SYBASE®
An SAP® Company

# Troubleshooting Guide

# Replication Server® 15.7.1 SP100

DOCUMENT ID: DC35920-01-1571100-01
LAST REVISED: May 2013

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

# Contents

# Conventions

These style and syntax conventions are used in Sybase® documentation.

*Style conventions*

| Key | Definition |
| --- | --- |
| `monospaced(fixed-width)` | • SQL and program code<br>• Commands to be entered exactly as shown<br>• File names<br>• Directory names |
| *`italic monospaced`* | In SQL or program code snippets, placeholders for user-specified values (see example below). |
| *italic* | • File and variable names<br>• Cross-references to other topics or documents<br>• In text, placeholders for user-specified values (see example below)<br>• Glossary terms in text |
| **bold san serif** | • Command, function, stored procedure, utility, class, and method names<br>• Glossary entries (in the Glossary)<br>• Menu option paths<br>• In numbered task or procedure steps, user-interface (UI) elements that you click, such as buttons, check boxes, icons, and so on |

If necessary, an explanation for a placeholder (system- or setup-specific values) follows in text. For example:

Run:

```
installation directory\start.bat
```

where *installation directory* is where the application is installed.

*Syntax conventions*

| Key | Definition |
|---|---|
| { } | Curly braces indicate that you must choose at least one of the enclosed options. Do not type the braces when you enter the command. |
| [ ] | Brackets mean that choosing one or more of the enclosed options is optional. Do not type the brackets when you enter the command. |
| ( ) | Parentheses are to be typed as part of the command. |
| \| | The vertical bar means you can select only one of the options shown. |
| , | The comma means you can choose as many of the options shown as you like, separating your choices with commas that you type as part of the command. |
| ... | An ellipsis (three dots) means you may repeat the last unit as many times as you need. Do not include ellipses in the command. |

*Case-sensitivity*

- All command syntax and command examples are shown in lowercase. However, replication command names are not case-sensitive. For example, **RA_CONFIG**, **Ra_Config**, and **ra_config** are equivalent.
- Names of configuration parameters are case-sensitive. For example, **Scan_Sleep_Max** is not the same as **scan_sleep_max**, and the former would be interpreted as an invalid parameter name.
- Database object names are not case-sensitive in replication commands. However, to use a mixed-case object name in a replication command (to match a mixed-case object name in the primary database), delimit the object name with quote characters. For example: **pdb_get_tables** "*TableName*"
- Identifiers and character data may be case-sensitive, depending on the sort order that is in effect.
  - If you are using a case-sensitive sort order, such as “binary,” you must enter identifiers and character data with the correct combination of uppercase and lowercase letters.
  - If you are using a sort order that is not case-sensitive, such as “nocase,” you can enter identifiers and character data with any combination of uppercase or lowercase letters.

*Terminology*

Replication Agent™ is a generic term used to describe the Replication Agents for Adaptive Server® Enterprise, Oracle, IBM DB2 UDB, and Microsoft SQL Server. The specific names are:

- RepAgent – Replication Agent thread for Adaptive Server Enterprise
- Replication Agent for Oracle

- Replication Agent for Microsoft SQL Server
- Replication Agent for UDB – for IBM DB2 on Linux, Unix, and Windows
- Replication Agent for DB2 for z/OS

# Troubleshooting Overview

A correctly configured Replication Server® system is designed to be fault-tolerant. However, in the event of a serious failure, you may need to intervene and manually fix the problem.

The key to finding the cause of a replication system failure is to eliminate possible causes by:

- Identifying recent tasks or events that might have adversely affected the replication system. A user may have performed a task that caused a replication failure or reduced the performance of the replication system. Many things can happen when a number of people are working on the same replication system.

  An event such as a temporary network outage may also have caused a replication system problem.
- Analyzing error logs.
- Determining the general problem category (configuration, subscription, replication, recovery).
- Using diagnostic tools, such as Replication Server programs or stored procedures, or **isql,** to analyze the replication system.

If you see an error message in a Replication Server error log, you can identify the problem by reading the error log. If there is no error message, use diagnostic tools to further analyze the replication system.

Replication Manager (RM) and the Embedded Replication Server System Database (ERSSD) problems are not included in the *Replication Server Troubleshooting Guide*. RM uses the Sybase Central™ message logging feature to provide a log of all commands sent by the RM to any server. It also has a view queue data feature that helps you troubleshoot transactions in a queue. See the online help for the Replication Manager plug-in and the *Replication Server Administration Guide Volume 1* for more information on how to use these features. See also the *Replication Server Administration Guide Volume 1* for more information on ERSSD recovery procedure.

The *Replication Server Troubleshooting Guide* may be able to help you to identify hardware, network, and operating system problems, but solving these problems is beyond the scope of the guide. Any time a server or a network connection is down, you should also check for hardware, network, or operating system problems.

On Windows, you can usually see a hardware or operating system problem when stack traces randomly or frequently occur at the same time you get errors in the Replication Server error log.

Check the operating system error log for errors that indicate hardware or operating system problems. Such failures might only partially resolve the effects on the replication system. You may still need to resynchronize data between the primary and destination databases.

# Tasks or Events That Can Cause Replication System Problems

Certain tasks or events can lead to a replication system failure. Check if any of these tasks have been performed or if the events have occured before attempting to categorize a problem.

**Table 1. Tasks or Events That Can Cause Replication System Problems**

| Task or Event | Description or Action |
|---|---|
| Binaries changed. | You might have changed binaries and have made the Replication Server, RepAgent, or Adaptive Server incompatible with each other. Incompatible binaries can produce Open Server™ and Open Client™ error messages. Check the latest release bulletins for version compatibility between replication system components. |
| Network went down and restarted. | Verify that Replication Server threads are up. |
| Rebuilt queues. | Check for manual recovery problems. |
| Recovered primary database. | Check for manual recovery problems. |
| Added, altered, or deleted routes. | The route operation might not be complete. Check for route problems. |
| Added, altered, or deleted subscriptions. | The subscription operation might not have completed. Check for subscription problems. |
| Added, altered, or deleted Replication Server, RepAgent, primary or replicate database, data server, or table. | You might have incorrectly changed the replication system. Check for configuration problems. |

### See also

- *Troubleshooting Manual Recovery Problems* on page 28
- *Troubleshooting Configuration Problems* on page 18
- *Subscription Problems* on page 75
- *Route Problems* on page 65
- *Verifying That Replication Server Threads are Up* on page 23

# Error Messages and Error Logs

Error messages provide important information for identifying the cause of replication system failures. When a Replication Server or RepAgent error occurs, an error message is recorded in an error log.

Error messages are in a standard format; once you learn this format, you will be able to read and understand all Replication Server and RepAgent error messages.

Replication Server error messages are recorded in text files called error logs, or sent to the standard error output (`stderr`), which is usually a terminal window. In general, Replication Agent error messages are recorded in their own error logs, except for RepAgent error messages. RepAgent records RepAgent errors and all error messages returned by Replication Server (including CT-Library error messages) in the Adaptive Server error log. Data server error messages are recorded in the data server's error log.

Multiple error messages can be generated by a single problem, and can be generated from any or all Replication Server components (including Adaptive Server, Replication Server, and RepAgent) that are adversely affected by the problem.

Replication Server and RepAgent can also print messages to the error log and to clients in several languages. Replication Server error messages appear in the language specified in the **RS_language** configuration parameter, while the Adaptive Server RepAgent uses the language specified in **language**.

**Note:** The *Replication Server Troubleshooting Guide* does not explain how to analyze the error logs from networks, gateways, non-Adaptive Server data servers, and non-Adaptive Server Replication Agents. See the documentation for these products for information about how to analyze their error logs.

### See also


- *Language, Sort Order, and Character Set Issues* on page 149


## Checking for Error Messages in the Error Logs

Check for and resolve error messages found in the error logs before checking for failures throughout your replication system.

1. Maintain open windows for all error logs, including:
   - Replication Server error logs (including a window for Replication Server standard error output (`stderr`))
   - Replication Agent logs (such as Replication Agent for DB2)
   - **rs_init** error log when setting up or configuring Replication Server.
   - Any data server error logs (such as Adaptive Server)

- Any gateway error logs (such as DirectConnect™ or OmniConnect™)

2. Scan the Replication Agent, data server, and gateway error logs to see if you can immediately find an error message that describes the cause of the error. If you find an error, see the documentation for the Replication Agent, data server, or gateway to solve the problem.
3. Scan the **rs_init**, Adaptive Server (for RepAgent error messages), and Replication Server error logs for error messages.

   - When you find an error, search for the error message in the *Replication Server Troubleshooting Guide*, and apply the documented workaround.
   - If you cannot find the error message, look in one of these error message text files in the `$SYBASE/$SYBASE_REP/doc` directory on UNIX platforms or the `%SYBASE%\%SYBASE_REP%\doc` directory on Windows:
     - `error_messages_rs` for Replication Server error messages
     - `error_messages_subcmp` for **rs_subcmp** error messages

     These text files contain the text of the error message, a brief description of the error, its cause, and, sometimes, how to fix it.

### See also

- *Common Error Messages* on page 35

## rs_init Error Log

Symptoms and problems are usually recorded in the **rs_init** error log when errors occur during Replication Server set up or configuration.

The **rs_init** error log is located in:

- UNIX: `$SYBASE/$SYBASE_REP/init/logs`
- Microsoft Windows: `%SYBASE%\%SYBASE_REP%\init\logs`

The **rs_init** error log file name format is:

```
logmonthday.session#
```

- *month* is a 2-digit integer that represents the current month.
- *day* is a 2-digit integer that represents the current day.
- *session#* is a 3-digit integer that represents the number of the session for that day. Each time a new **rs_init** session is started on the same day, the session number is increased by one.

Given that `log0108.001` is an **rs_init** error log:

- *month* is 01
- *day* is 08
- *session#* is 001

## Replication Server Error Log

Informational, warning, thread-terminated, fatal, and internal error messages from Replication Server are recorded in the Replication Server error log.

Messages are appended to the error log while Replication Server is operating. Error messages are appended to the end of the existing error log each time you restart Replication Server.

**Warning!** Real-time signals 32-64 terminate Replication Server without any error message in the log.

Informational messages report the current status of a component; for example, a process or command has completed or a component has been shut down. Warning, thread-terminated, fatal, and internal error messages are reported when a component abnormally terminates, a process or command cannot be completed, or a fatal internal error occurs in a component.

When Adaptive Server errors cause a Replication Server error, you find references to the Adaptive Server errors in the text of the Replication Server error message. In these cases, you may fix the Adaptive Server problem, which should eliminate the Replication Server error.

The Replication Server error log may also contain Open Client/Server™ error messages, since Replication Server is an Open Server program with Open Client components. Any Open Server errors in the error log constitute internal errors.

Replication Server error logs contain error messages generated during the execution of asynchronous commands, such as **create subscription** and **create route**. When you are executing asynchronous commands, pay special attention to the error logs for the Replication Servers affected by the asynchronous command.

Dedicate a terminal window to show the standard error output from a Replication Server. If the Replication Server error log is unavailable, error messages are sent to `stderr`. Error messages that are sent only to `stderr` are extremely serious, because the only action available to the Replication Server is to dump the stack and exit.

The Replication Server error log file name format is:

```
rs_name.log
```

where *rs_name* is the Replication Server name.

The default location for the Replication Server error log is:

- UNIX: `$SYBASE/$SYBASE_REP/install`
- Microsoft Windows: `%SYBASE%\%SYBASE_REP%\install`

## Analyzing the Replication Server Error Log

Analyze the Replication Server error log to determine the cause of the replication system failure.

To analyze a Replication Server error log, you must understand how multiple error messages are related. Multiple error messages can be generated by the same problem and usually have identical, or very close, dates and times. Error messages that occur close together constitute an error block.

1. Scan from the end of the error log to the beginning, looking for the most recent Replication Server error. To identify the problem, find the error block to which the error belongs.

   As Replication Server errors occur, Replication Server appends the error messages to the end of the error log. The most recent Replication Server error marks the end of the error block.

   **Note:** If you see an Open Server/Client error such as `no more threads`, fix that error first. This message means you have run out of an Open Server resource (and Replication Server might have terminated).

2. Find and diagnose the last non-Replication Server error message that occurred after the first Replication Server error. If there are no non-Replication Server error messages after the first Replication Server error message, investigate the causes of the first Replication Server error message.

   In diagnosing non–Replication Server errors, you may need to use other troubleshooting or error message guides—such as those for Adaptive Server or Open Client (particularly with network communication or other client problems).

3. In the error block, find any Replication Server error messages that tell you if a Replication Server component or thread has terminated.

   You must restart the component or thread that has terminated. Usually, subsequent Replication Server errors in the same error block were caused by the first Replication Server error.

## Replication Server Error Message Format

Describes the Replication Server error message format.

The format of Replication Server error messages is:

```
s. date time severity_text error_num thread
(thread_context) module (line_num) error_text
```

where:

- *s* – a letter that indicates the severity of the error.
- *date* – date.
- *time* – time the error occurred, in 24-hour format.

- *severity_text* – brief phrase that describes the severity.
- *error_num* – integer that uniquely identifies the error. Replication Server error numbers are constructed using:

  *module_number* * 1000 + *error_condition*

  A *module_number* is assigned to each source code module in the server and the *error_condition* is a numbered error condition within the source code module.
- *thread* – name of the Replication Server thread that received the error.
- *thread_context* – more specific information about the thread, such as the database and data server if it is a Data Server Interface (DSI) thread.
- *module* – name of the Replication Server source file where the error was reported. This name is used only by Sybase Technical Support.
- *line_num* – line in the Replication Server source file where the error was reported. This number is used only by Sybase Technical Support.
- *error_text* – error message text. If a Replication Server error was caused by an Adaptive Server error, the Adaptive Server error is included as part of the Replication Server error message text.

## Severity Code Reference

Replication Server error messages begin with a letter that indicates the severity of the error.

**Table 2. Replication Server Error Severities**

| Severity Code | Description |
|---|---|
| I | An informational message. Error messages with a severity of "I" contain only the date and time the error occurred, and the full text of the message. |
| W | A warning about a condition that has not yet caused an error, but may require attention, for example, running out of a resource. |
| E | An error that does not prevent further processing, such as a site that is unavailable. |
| H | A Replication Server thread has died, for example, a lost network connection. |
| F | Fatal. A serious error caused Replication Server to exit. For example, if you start the Replication Server using an incorrect configuration, you are likely to receive a fatal error. |
| N | Internal error, caused by anomalies in the Replication Server software. Report these errors to Sybase Technical Support. |
| T | A trace message. |

For more information about error severity, see the *Replication Server Administration Guide Volume 2*.

## Example: Analysis of a Replication Server Error

Examines a Replication Server error message block.

Replication Server error message block:

```
H. 2006/06/15 20:52:28. THREAD FATAL ERROR #5049 DSI
EXEC(104(3) WESTERNDS.westDB) - dsiqmint.c(3252) The
DSI thread for database 'WESTERNDS.westDB' is being
shutdown. DSI received data server error #102 which is
mapped to STOP_REPLICATION. See logged data server
errors for more information. The data server error was
caused by output command #1 mapped from input command
#2 of the failed transaction.
I. 2006/06/15 20:52:28. The DSI thread for database
'WESTERNDS.westDB' is shutdown.
```

**Table 3. Analysis of the Replication Server Error Message**

| Error Message Text | Explanation |
|---|---|
| H. 2006/06/15 20:52:28. THREAD FATAL ERROR #5049 DSI EXEC(104(3) WESTERNDS.westDB) - dsiqmint.c(3252) | This is the first error. The H identifies the error as a thread termination error. This message shows that the Data Server Interface (DSI) thread terminated.<br>The error message includes the dataserver.database name where the error occurred, the internal Replication Server module (`dsiqmint.c`), and the line of code (3252) where the error occurred. |
| The DSI received data server error # 102 which is mapped to STOP_REPLICATION. See the logged data server errors for more information. | This Adaptive Server error message identifies the cause of the problem.<br>Find a description for Adaptive Server error 102 by:<br>• Finding the error in the Adaptive Server error log (the error would occur at approximately the same time as it occurred in the Replication Server error log),<br>• Performing a `select * from sysmessages where error = 102` in the master database, or<br>• Looking up the number in the *Adaptive Server Enterprise Troubleshooting and Error Messages Guide*.<br>The *Adaptive Server Enterprise Troubleshooting and Error Messages Guide* describes error 102 as invalid syntax near %s for a Transact-SQL® statement. It also provides suggestions about what might be causing this error (for example, a keyword is spelled incorrectly, a keyword or parameter is missing, or the order of the keyword is incorrect). Look for additional information in the data server error log, which includes the character string for the %s field in the error message. |

| Error Message Text | Explanation |
|---|---|
| The data server error was caused by RS output command #1 mapped from input command #1 of the failed transaction. | This text describes the command position in a grouped transaction sent by the Replication Server. |
| I. 2006/06/15 20:52:28. The DSI thread for database ‘WESTERNDS.westDB' is shutdown. | This last error message is informational (I) and caused by the problem specified in a previous block. After fixing the Adaptive Server problem, restart the DSI thread for the specified database. |

## RepAgent Error Message Format

RepAgent error messages are recorded in the Adaptive Server error log using the Adaptive Server message format. These messages are identified by the string “RepAgent(*dbid*),” which appears in the first line of the message. Errors that you can retry are logged only once in the Adaptive Server error log.

The RepAgent error message format is:

```
date time RepAgent (dbid): error_number, severity,
state, error_text
```

where:

- *date* – date that the error occurred.
- *time* – time that the error occurred.
- *dbid* – Adaptive Server identification number of the database that RepAgent is using. You can find this database ID by executing:

  ```
  select x = db_id()
  ```
- *error_number* – RepAgent error message numbers range from 9200 to 9299.
- *severity* – severity can be:
  - EX_INFO – informational error message.
  - EX_USER – user error.
  - EX_RESOURCE – resource error in which an operating system resource or Replication Server resource is not available.
  - EX_CMDFATAL – fatal error in which RepAgent cannot continue processing a transaction.
- *state* – for internal use only.
- *error_text* – description of the cause of the error.

## Example: Analysis of a RepAgent Error Message

Examines a RepAgent Error message.

RepAgent error message:

```
00:00000:00036:2006/01/13 13:08:16.39 server  Error:
9209, Severity: 20, State: 0
00:00000:00036:2006/06/23 13:08:16.39 server
```

```
RepAgent(6): Missing datarow in TEXT/IMAGE insert log
record. Transaction log may be corrupt. Please contact
SYBASE Technical Support. (current marker = (107634,
10)).
```

**Table 4. Analysis of the RepAgent Error Message**

| Error Message Text | Explanation |
|---|---|
| 2006/01/13 | Date |
| 13:08:16.39 | Time |
| (6) | Database ID |
| 9209 | Error number |
| 20 | Severity |
| 0 | State |
| Missing data row in TEXT/IMAGE insert log record. Transaction log may be corrupt. Please contact SYBASE Technical Support. (current marker = (%d, %d)). | Error message text |

For more information about the Adaptive Server error message format, see the *Adaptive Server Enterprise System Administration Guide*.

# Types of Replication System Problems

The types of problems that occur in a replication system roughly correspond to the different stages of development of the replication system. A replication system consists of Replication Server components (Replication Servers, Replication Agents, data servers, routes, connections) connected together such that data is copied reliably from source tables to destination tables.

## Configuration Problems

Configuration problems occur during Replication Server set-up, such as when Replication Servers, RepAgents, and data servers are added to the replication system using **rs_init**.

Usually, symptoms and problems are identified by error messages in the **rs_init** log file.

Some configuration problems cause subscription materialization failures, and their symptoms do not appear until you attempt to materialize a subscription.

### See also

- *Troubleshooting Configuration Problems* on page 18
- *rs_init Error Messages* on page 35

## Subscription Problems

Subscription problems occur when subscription materialization or dematerialization fails.

The replication process begins with subscription materialization, which is the process by which data is initially copied to the destination database. When you no longer want a subscription replicated to a destination database, dematerialize the subscription at the destination database. Dematerialization is the process by which data is deleted from the destination database.

**Note:** If you are using a Replication Agent, your subscription materialization process may differ from the process described here. See your Replication Agent documentation for the Replication Agent-specific subscription materialization process.

Subscription problem symptoms are easily identified and include:

- Materialization failure – no data in the subscription's replicate table at the destination database, invalid status for subscriptions at the primary and replicate Replication Servers, or materialization has been taking longer than is reasonable.
- Dematerialization failure – data still exists in the subscription's replicate table at the destination database, the status for subscriptions at the primary and replicate Replication Servers is invalid, or dematerialization has been taking longer than is reasonable.

Usually, the person who is conducting the materialization or dematerialization monitors the operation and reports any problems.

Some subscription problem symptoms are reported as error messages in the Replication Server error log. You might also need to use the diagnostic tools to identify subscription problem symptoms.

If a subscription problem caused the Data Server Interface (DSI) thread for the replicate database to abnormally terminate, restart it using the **resume connection** command.

### Function String Restrictions

Customized function strings can be used to replicate changes. Incorrect variables can cause problems in customized function strings.

Function string restrictions include:

- Only function strings for **rs_insert** and **rs_update** can use new column values.
- Only function strings for **rs_delete** and **rs_update** can use old column values.
- Only function strings for **rs_select** and **rs_select_with_lock** can have input templates, and use user-defined variables.
- Only function strings for user-defined functions can use parameter values of functions. The parameter value of a function consists of the parameters passed to a replicated stored procedure.

## Replication Problems

Replication problems occur when data changes at the primary database are not applied to the destination database.

Replication consists of copying data operations, such as updating or deleting data, from a primary database to the destination database. Replication begins after a subscription has been successfully materialized.

If you are monitoring the replication system, you might directly observe that data is not replicating to a destination database. Use **rs_subcmp** to determine which subscription is not being replicated.

If someone reports that their client application has retrieved incorrect data from a destination database consider that a replication problem may exist. Compare the primary and destination tables; if they are the same, then data is being replicated correctly, and it is likely that a problem with the client application that is causing incorrect data to appear in the client application. If data is not the same at the primary and destination databases, replication is failing, and you must troubleshoot the replication system.

Some symptoms of a replication problem directly identify the cause; other symptoms require more investigation to find the underlying cause. These symptoms are listed in order of most common to least common:

- Data Server Interface (DSI) thread is down.
- Threads other than DSI are down.

  Use **admin who_is_down** to display information about threads that are down:
  - DIST (Distributor) thread
  - RepAgent user thread
  - RSI (Replication Server Interface) thread
  - RSI user thread
  - RS (Replication Server) user thread
  - SQM (Stable Queue Manager) thread
  - SQT (Stable Queue Thread) thread
  - NRM (Normalization) thread
  - Major replication system component is down.

  Use **isql** to check to see if a server is down by logging in to each server.
  - RepAgent
  - Replication Server
  - Data server
- A `detecting loss` message, which means that data replication messages were lost after queues were rebuilt. This message is shown in the Replication Server error log or in the `rs_oqid` system table.

See the Loss Status column in the output from **admin health** and **admin who, sqm** to monitor data consistency and check if there is the possibility of data loss in the queues.

- Inbound or outbound stable queues are growing larger.

  Use **admin who, sqm** and **sysadmin dump_queue** to display information about inbound and outbound stable queues.
- Number of duplicate transactions is increasing.

  Use **admin who,sqt** and **sysadmin dump_queue** to display information about inbound and outbound stable queues.
- Transactions remain open for longer than is reasonable. These transactions might be orphans or a very long transaction. Orphan transactions do not have an ending **commit** or **rollback** statement.

  Use **admin who, sqt** and **sysadmin dump_queue** to display information about inbound and outbound stable queues.
- Primary and destination Replication Servers do not have the same locater.

  Use **isql** to log in to the RSSD and view the `rs_locater` system table.
- Replication is successful for other subscriptions on different data servers with connections to the same destination Replication Server.

  Use **rs_subcmp** to compare a subscription's tables in the primary and replicate databases to make sure the tables are the same.
- Replication is successful for other subscriptions in the same or different tables on the same data server while replication for a particular subscription is failing.

  Use **rs_subcmp** to compare a subscription's tables in the primary and replicate databases to make sure the tables are the same.

Some symptoms appear as error messages in the Replication Server and Adaptive Server error logs. Use the diagnostic tools to identify replication problem symptoms.

### See also

- *Troubleshooting Replication Failures* on page 21
- *Diagnostic Tools* on page 18

## Manual Recovery Problems

Manual recovery problems occur during the recovery of a failed partition, truncated primary database log, primary database failure, or Replication Server System Database (RSSD).

While Replication Server is designed to tolerate most failures and to recover from them automatically, some failures require you to intervene by manually executing recovery tasks. Sometimes after you have completed a recovery task, you run into replication problems or errors show up in the Replication Server error log. Although the *Replication Server Administration Guide Volume 1* and *Replication Server Administration Guide Volume 2* discuss most recovery situations and you can refer to those manuals to see if you missed a step in a recovery task, the *Replication Server Troubleshooting Guide* dicusses the most common

problems that you might run into after you think that you have successfully completed a recovery task.

### See also

- *Troubleshooting Manual Recovery Problems* on page 28

## Diagnostic Tools

Diagnostic tools retrieve the status and statistics of Replication Server components. Depending on the type of problem, use these status and statistics to analyze the replication system.

- **isql** – use to log in to a Replication Server or data server to see if servers are running. You can also use **isql** to execute SQL commands to see if data is the same in the primary and replicate databases, or if data has been materialized or dematerialized.
- **admin who_is_down** – identifies which Replication Server threads are down.
- **admin who, sqm** – displays information about stable queues at a Replication Server, such as the number of duplicate transactions, the size of stable queue, and the data loss status.
- **admin who, sqt** – displays information about stable queues at a Replication Server such as the number of open transactions.
- **admin stats, md** – displays information about messages delivered by a Replication Server such as the number of messages delivered.
- **sp_config_rep_agent** – displays the current RepAgent configuration settings.
- **sp_help_rep_agent** – displays static and dynamic information about a RepAgent thread.
- **sysadmin dump_queue** – dumps stable queues and lets you view them.
- **rs_helproute** – displays the status of routes at a Replication Server.
- **rs_subcmp** – compares a subscription's tables in the primary and replicate databases. Use **rs_subcmp** to make sure the tables are the same.
- **check subscription** – displays the status of subscriptions at a Replication Server.
- **rs_helppub** – displays publications.
- **rs_helppubsub** – displays publication subscriptions.
- **sp_setrepcol** – checks the replication status of `text`, `unitext`, or `image` columns.

## Troubleshooting Configuration Problems

Verify that a replication system is configured correctly by materializing subscription data. Some configuration problems do not appear until you attempt to materialize subscription data.

The most common configuration problems that cause materialization failures are:

- Failure to log in to the primary Adaptive Server. The user who creates the subscription in the replicate Replication Server must have the same login name and password both in the primary Adaptive Server and the primary Replication Server.
- Missing permissions in the primary database. The user who creates the subscription must be a user in the primary database and must have **select** permission in the primary table.
- Missing permissions in the replicate database. The maintenance user must have **select**, **insert**, **update**, and **delete** permissions on the tables in the replicate database.
- A Replication Server or Adaptive Server has stopped running. Try to log in to each server using **isql**. Restart any servers that are not running.

Other common configuration problems include:

- Host name resolution error.
- The Adaptive Server entry does not exist in the interfaces file.

For more information about troubleshooting Replication Server configuration problems, see the *Replication Server Configuration Guide* for your platform.

### See also

- *rs_init Error Messages* on page 35

## Troubleshooting Materialization Failures

Troubleshoot subscriptions that have not materialized.

### Prerequisites

Verify that data has failed to materialize by logging in to the replicate database using **isql** and executing a **select** command that selects the materialized columns from the replicate table.

Also make sure that all tasks required for subscription materialization have been completed. Subscription materialization may fail if you have not completed:

- Creating replication definitions or function replication definitions
- Marking tables or stored procedures for replication
- Creating connections to the destination databases
- Creating articles, if you are using them
- Creating and validating publications, if you are using them
- Marking `text`, `unitext`, or `image` columns for replication, if you are replicating `text`, `unitext`, and `image` columns.
- Creating direct and indirect routes, if the destination database is connected to a Replication Server different from the primary database's Replication Server
- Creating logical connections, if you are using warm standby applications

### Task

1. If you are materializing a large amount of data, ensure that the *num_threads* and *num_concurrent_subs* parameters are large enough.
2. Log in to the destination Replication Server and issue **check subscription**, which returns information that diagnoses the problem, including:
   - Other subscriptions to the same replication definition and replicate database have not yet processed
   - No connection to the primary Replication Server because of an incorrect login
   - Primary Replication Server down or out of stable queues
   - Stable Queue Manager (SQM), Stable Queue Transaction interface (SQT), and Distributor (DIST) threads down
   - Primary data server down, incorrect login, out of stable queues, or rows selected with holdlock
   - RepAgent problem
   - Route problem
   - Destination Replication Server—incorrect login or out of stable queues
   - Destination Replication Server Data Server Interface (DSI) problem—use **admin who, dsi** or **admin who, sqm** to determine what the specific problem is
   - Incorrect user privileges on destination database
3. Log in to the primary Replication Server and also check for its subscription status using **check subscription**.
4. Use **rs_helppub** and **rs_helppubsub** to find the publications and articles that a subscription is using.
5. If some columns are not being materialized:
   a) Check replication status of `text`, `unitext`, and `image` columns.
   b) Verify that the replication definition is correctly defined.
   c) Verify that the publications and articles are correctly defined.
6. Fix the problem.
7. Run the replication system when you think you have solved the problem.

   If the subscription is still not materialized, analyze the error log again or complete any of the steps you have skipped.

## Troubleshooting Dematerialization Failures

Troubleshoot subscriptions that have not dematerialized.

### Prerequisites

Verify that data has failed to dematerialize by logging to the replicate database using **isql** and executing a **select** command that selects the dematerialized columns from the replicate table.

### Task

1. Log in to the destination Replication Server and issue **check subscription**, which returns information that diagnoses the problem, including:
   - Other subscriptions to the same replication definition and replicate database have not yet processed
   - No connection to the primary Replication Server because of an incorrect login
   - Primary Replication Server down or out of stable queues
   - Stable Queue Manager (SQM), Stable Queue Transaction interface (SQT), and Distributor (DIST) threads down
   - Primary data server down, incorrect login, out of stable queues, or rows selected with holdlock
   - RepAgent problem
   - Route problem
   - Destination Replication Server DSI problem
   - Incorrect user privileges on destination database
2. Log in to the primary Replication Server and also check its subscription status using **check subscription**.
3. If some columns are not being dematerialized:
   a) Check replication status of `text`, `unitext`, and `image` columns.
   b) Verify that the replication definition is correctly defined.
4. Fix the problem.
5. Run the replication system when you think you have solved the problem.

   If the subscription is still not dematerialized, analyze the error log again or complete any of the steps you have skipped.

## Troubleshooting Replication Failures

Isolate replication failures, which may occur after subscriptions have successfully materialized.

Before you troubleshoot a replication failure, verify that data is not replicating. The troubleshooting procedures listed here can be executed separately and in any order; however, they are listed in the order in which they are most likely to solve the problem. These procedures assume that the replication system has:

- Been installed and configured correctly,
- Successfully completed subscription materialization, and
- Correctly replicated data previousl.y

After executing the procedures, run the replication system to check if the problem has been solved. If replication is still not executing correctly:

- Check the error logs for error messages.
- Perform any of procedures that you skipped.

## Verifying That Data is Not Replicating

Verify that data is not replicating before troubleshooting a replication failure.

1. Log in to the primary and replicate databases using **isql**.
2. Execute **select** commands that select the replicate columns from the replicate table and the columns to be replicated from the primary table.
3. Compare the data from both tables to see if it is the same.

   Incompatible data means that the system is not replicating properly. If the subscription has many columns that are replicated, use **rs_subcmp** to compare data in the primary and replicate databases.

## Identifying the Data That is Failing to Replicate

Determine the specific subscriptions and columns that are failing to replicate.

This also verifies that the primary and destination data servers, and the primary or destination Replication Server are running.

1. Use **isql** to log in to the primary or destination Replication Server.
   If you cannot log in to a Replication Server, then it is down.
2. Run **rs_subcmp** to find out which data in a suspect subscription is failing to replicate.

   **rs_subcmp** logs in to the primary and destination data servers and compares the subscription's data in the primary and destination tables. **rs_subcmp** can compare tables at Adaptive Server data servers only. To compare tables at a non-Adaptive Server data server, you can use a program equivalent to **bcp out** on the non-Adaptive Server data server and **bcp out** on the Adaptive Server data servers, then use the UNIX **diff** command to compare the output.

   - If **rs_subcmp** displays inconsistent rows, note the columns and rows that are not being replicated.
   - If only `text`, `unitext`, and `image` columns are not being replicated, these columns may have inconsistent replication status.
   - If no data exists for subscribed columns, the subscription has not materialized.
   - If **rs_subcmp** fails, one or both of the data servers are down:
     - If the primary data server is down, the Adaptive Server log may be corrupt or full. The data server may also have an operating system or hardware error.
     - If the destination data server is down, there may be a Data Server Interface (DSI) problem, or an operating system or hardware error.
3. Use **rs_subcmp** to check if other subscriptions on the same data server are replicating:

- If no other subscriptions are replicating, it is likely that a problem exists with that data server rather than with a particular subscription.
- If all other subscriptions are replicating, then a problem may exist with that particular subscription.

4. Use **rs_subcmp** to check if other subscriptions on databases controlled by the same destination Replication Server are replicating. If replication is working for other databases controlled by the destination Replication Server, then the problem is a specific database, database connection, or RepAgent. Perform these:

   - Look for orphaned transactions in the primary Replication Server inbound queue for the database.
   - Troubleshoot RepAgents.
   - Troubleshoot database connections.

### Next

Once you have identified the data that is failing to replicate, verify that the Replication Server threads are up.

### See also

- *Replication Server Is Down* on page 25
- *Adaptive Server Log Problems* on page 127
- *Data Server Interface Problems* on page 117
- *RepAgent Problems* on page 107
- *Error 32046* on page 113
- *Troubleshooting Materialization Failures* on page 19
- *Checking for Orphaned Transactions* on page 130

## Verifying That Replication Server Threads are Up

Use **admin who_is_down** to determine if the primary and destination Replication Server threads are up.

1. Log in to the destination Replication Server.
   If you cannot log in to a Replication Server, then it is down.
2. Execute **admin who_is_down**.

   This command displays all the threads that are down on this Replication Server, and records error messages in the Replication Server error log.
3. Log in to the primary Replication Server and use **admin who_is_down** to display all the threads on this Replication Server that are down.
   a) Check the Replication Server error log for these conditions:
      - The Data Server Interface (DSI) is down,

- The RepAgent is not connected to the Replication Server or Adaptive Server, and
- The entire (or part of the) network went down and was restarted.

If these conditions exist, it indicates that the *keepalive* value is set too low and that the TCP connection was terminated and never restarted.

4. If the DSI is up, check for data loss.

   Data loss error messages do appear in the Replication Server error log, however, these errors only show up once and may have occurred several days earlier.

### Next

If a thread is down, determine the cause of the failure and correct the problem.

| Thread That Failed | Action |
|---|---|
| Distributor (DIST) | Determine if the failure is due to Replication Server error 7035 or 13045, and correct the problem. |
| DSI | May indicate duplicate keys or permission failure, see *Replication Server Troubleshooting Guide > Data Server Interface Problems*. |
| DSI EXEC | See *Replication Server Troubleshooting Guide > Data Server Interface Problems*. |
| RepAgent User | See *Replication Server Troubleshooting Guide > RepAgent Problems*. |
| Replication Server (RS) User | See *Replication Server Troubleshooting Guide > Subscriptions Problems*. |
| Replication Server Interface (RSI) and RSI User | See *Replication Server Troubleshooting Guide > Replication Server Interface Problems*. |
| Stable Queue Manager (SQM) | The SQM should not go down. Restart the Replication Server; you cannot resume the SQM thread. |
| Stable Queue Thread (SQT) | Determine if the failure is due to Replication Server error 13045, and correct the problem. |
| User | This should have no effect on replication. |

### See also

- *Data Server Interface Problems* on page 117
- *Replication Server Interface Problems* on page 103
- *RepAgent Problems* on page 107

- *Subscription Problems* on page 75
- *Error 13045* on page 50
- *Error 7035* on page 46

## Replication Server Is Down

If the primary, destination, or both Replication Servers are down, analyze the error log for each server.

If both Replication Servers are down, the primary and replicate Replication Servers may each have different problems. If both Replication Servers have their Replication Server System Databases (RSSDs) on the same Adaptive Server data server, troubleshoot the Adaptive Server data server.

## Checking for Queue Problems

Determine whether there is an increase in the number of duplicate transactions.

1. Run **admin who, sqm** to see if the number of duplicate transactions is increasing.

   The duplicate count increases when the Data Server Interface reads a transaction that has already been applied to the replicate Replication Server.
2. If the duplicate count is increasing, check the outbound queues for stuck and open transactions.

### See also

- *Data Server Interface Problems* on page 117

## Verifying That All RepAgents are Up

Use **sp_who** to view the Adaptive Server RepAgent thread status.

1. Log in to the primary Adaptive Server data server using **isql**.
2. Run **sp_who** to verify that the RepAgent is active.

   See *Adaptive Server Enterprise > Reference Manual: Procedures > System Procedures > sp_who*.

   If the RepAgent is down, one of these can be the cause of its failure:
   - The Adaptive Server log is corrupt.
   - The Adaptive Server log is full.
   - The RepAgent set an illegal truncation point.
   - The RepAgent used an incorrect primary Replication Server login.
   - `text`, `unitext`, or `image` columns have inconsistent replication status.
   - Incorrect replication system configuration.

### See also

- *RepAgent Problems* on page 107
- *Adaptive Server Log Problems* on page 127

## Checking System Tables

Check the Replication Server System Database (RSSD) tables for error information.

1. Identify any recoverable actions in the `rs_recovery` system table and perform the actions specified.
2. Look for any `detecting losses` or `rejecting messages after loss detected` validation status messages in the `rs_oqid` system table. (You might have missed the message in the error log.)

   A `detecting loss` message signifies that data replication messages were lost after queues were rebuilt.

   See the Loss Status column in the output from **admin health** and **admin who, sqm** to check if there is the possibility of data loss in the queues.

### See also

- *Detecting Loss for Database* on page 56

## Finding Failed Replication Component

To determine the component on which replication has stopped, turn on traces and examine the output of **sysadmin dump_queue**.

1. Specify trace flags when you run the diagnostic versions of the Replication Server (`repserver.diag`).
2. Run **sysadmin dump_queue** and examine its output.
   If replication stopped in one of the components, determine the cause of failure and rectify the problem.

| Component | Troubleshooting Reference |
|---|---|
| Primary data server | • *Replication Server Troubleshooting Guide > Data Server Interface Problems*<br>• *Replication Server Troubleshooting Guide > Adaptive Server Log Problems* |
| RepAgent | *Replication Server Troubleshooting Guide > RepAgent Problems* |

| Component | Troubleshooting Reference |
|---|---|
| Primary data server | • *Replication Server Troubleshooting Guide > Data Server Interface Problems*<br>• *Replication Server Troubleshooting Guide > Adaptive Server Log Problems* |
| Outbound queue | • *Replication Server Troubleshooting Guide > Route Problems*<br>• *Replication Server Troubleshooting Guide > Data Server Interface Problems* |
| Data Server Interface (DSI) thread | *Replication Server Troubleshooting Guide > Data Server Interface Problems* |
| Custom function-string class or customized function strings | • *Replication Server Troubleshooting Guide > Data Server Interface Problems*<br>• *Replication Server Troubleshooting Guide > Stable Queues > Using Traces to Print Commands* |

### See also

- *Data Server Interface Problems* on page 117
- *Adaptive Server Log Problems* on page 127
- *RepAgent Problems* on page 107
- *Route Problems* on page 65
- *Using Traces to Print Commands* on page 140

## Checking for Route Problems

Use **rs_helproute** to check for route problems.

1. Make sure no other databases are replicating through the same route as the non-replicating subscription.

   If other databases are replicating through that route, then problems within the primary Replication Server/database and replicate Replication Server/database are more likely. It is unusual for replication to have stopped for one database while replication continues through the same route for other databases. Run **admin stats, md** on the replicate Replication Server to see if "Messages Delivered" increases over time.
2. If no other databases are replicating through the same route as the nonreplicating subscription, run **rs_helproute** at the primary or destination Replication Server in the RSSD to determine if there is a route problem.

### See also

- *Route Problems* on page 65

## Troubleshooting Manual Recovery Problems

Manual recovery problems may occur during recovery of a primary database.

If you recently recovered a primary database and replication is failing, you may be seeing one of these problems:

- A new generation ID for the primary database has not been set. See *Replication Server Administration Guide Volume 2*.
- If you loaded the primary and replicate databases from the same backup, make sure the `rs_lastcommit` table for the replicate database has the correct entries. If the *origin_qid* in the `rs_lastcommit` table is incorrect (old or changed), the Data Server Interface (DSI) may ignore certain transactions associated with that *origin_qid*. Use **bcp out** to copy data from the `rs_lastcommit` table, load the data, and then copy the data back to `rs_lastcommit` using **bcp in**. If DSI still ignores certain transactions, update the *origin_qid* value to 0 to force DSI to accept these transactions.
- If you used dump and load to synchronize the primary and replicate databases because of a failure, increase the generation number in the primary database. If you do not correctly set the generation number, replication from this database may stop because Replication Server perceives that it has already processed the messages.

See *Replication Server Administration Guide Volume 2* for information about recovering primary databases.

# Performance Problems

Reduced performance usually occurs when the operating system or machines are overloaded as a result of increased demands on the system. Reduced performance can result from adding applications or Replication Server components, executing very large transactions, or even upgrading the operating system.

Performance problems can be critical or noncritical. You can eliminate noncritical performance problems by optimizing your replication system. For noncritical problems, see the *Replication Server Administration Guide Volume 2*.

If critical performance problems are left unresolved, performance degradation can lead to fatal problems, such as full stable queues, in which replication stops. A replication, materialization, or dematerialization failure can be caused by a critical performance problem.

Critical performance problems may be caused by:

- Introducing new components, such as Adaptive Servers, databases, Replication Servers, RepAgents, or Replication Agents. New components may cause resource contention and overload any component.
- Changing the operating system. Upgrading the operating system, applying patches, changing kernel parameters, or rebuilding the kernel may adversely impact your replication system, memory allocation, and resources.
- Adding applications to the replication system, which may impact memory requirements and use resources.
- Replicating a very large database, which may produce a very high latency. Large transaction or an open transaction is also a possible cause.

## Insufficient Stable Queue Size

Degraded Replication Server performance can sometimes be caused by the Stable Queue Transaction (SQT) cache size being too small.

If the cache is not big enough to hold all open transactions, transactions that cannot completely fit into the cache are processed one command at a time. The inbound stable queues grow because Replication Server cannot process the transactions fast enough.

*Solution*

Increase the stable queue size to process more transactions and improve Replication Server performance:

1. Make sure that the problem is not caused by orphaned transactions.

2. Find the optimal size for the *sqt_max_cache_size* parameter for your application. See *Replication Server Administration Guide Volume 2*.
3. Suspend the connection to the data server.
4. Log in to Replication Server and run:

```
> configure replication server
   set sqt_max_cache_size to 'new_value'
> go
```

5. To activate the new values, resume the connection. To dump the SQT cache, use **sysadmin sqt_dump_queue** command.

### See also

- *Checking for Orphaned Transactions* on page 130

## Reduced Performance when Replicating to Sybase IQ

When using real-time loading (RTL) replication, **INSERT ... LOCATION** execution failures can slow down replication to Sybase IQ.

Sybase IQ must connect to Replication Server to retrieve data when Replication Server sends an **INSERT ... LOCATION** statement to Sybase IQ. If Sybase IQ fails to connect, the **INSERT ... LOCATION** statements are not executed and RTL fails. After several unsuccessful attempts to replicate using RTL, Replication Server reverts to log-order, row-by-row continuous replication.

*Solution*

Create an entry for the replicate Replication Server in the `interfaces` file of the replicate Sybase IQ server. This allows Sybase IQ to connect to Replication Server using the user name and password specified in the database connection. The maintenance user must be a valid user in Replication Server with system administrator privileges and whose password matches the password that Replication Server uses to log in to Sybase IQ.

## Unable to Continue Replication in a Faster Mode

To determine the cause of replication failure, suspend replication before Replication Server retries to apply transactions, or reverts to the continuous log-order language replication mode.

Replication Server may fail to apply transactions to the replicate database if Replication Server encounters limitations, processing errors, or thresholds for faster replication modes (high-volume adaptive replication (HVAR), real-time loading (RTL), dynamic SQL, parallel DSI, or DSI bulk copy-in). Replication Server retries to apply transactions and automatically switches to the continuous log-order language replication mode if Replication Server cannot continue to use the faster modes, and even after trying to reapply transactions in smaller compilable groups in HVAR and RTL modes.

When Replication Server fails to apply transctions in a replication mode other than the continuous log-order language mode, replication performance may be affected. You can use commands such as **admin stats** and **admin stats, {tps | cps | bps}** to check replication performance.

To identify when Replication Server is failing to replicate using any of the replication modes, and to investigate replication performance problems, configure the **dsi_retry** parameter to suspend the DSI thread once the failure occurs and before Replication Server switches to the continuous log-order language mode from one of the other modes. When **dsi_retry** suspends the DSI thread, Replication Server cannot apply transactions to the replicate database, and you can analyze the Replication Server log file or dump queue for information about the failed transactions to determine the reasons for the failure to apply transactions. For example, you may see in the `repserver.log` file an entry such as:

```
I. 2012/09/11 19:53:03. A grouped transaction of 5 individual
transactions has failed in database 'repl4_28382.rdb' with 'HVAR/
RTL'.
DSI is now suspended because dsi_retry value '2'.
E. 2012/09/11 19:53:03. ERROR #1028 DSI EXEC(104(1) repl4_28382.rdb)
- dsiqmint.c(4471)
        Message from server: Message: 546, State 1, Severity 16 --
'Foreign key constraint violation occurred, dbname =  'rdb', table
name = 'tb15_1', constraint name = 'tb15_1_col_1152004104'.
```

This is an error message that Replication Server generates when HVAR is applying **update** statements to a table with a referential constraint. Unless you use replication definitions to specify tables with referential constraints, the HVAR and RTL process to apply updates cannot avoid referential integrity constraints failures, and HVAR and RTL continue to try to apply the transactions to the replicate database using progressively smaller transaction groups. Eventually, Replication Server switches to the slower continuous log-order language replication mode when attempts with HVAR and RTL fail, resulting in an adverse impact on replication performance. To fix the HVAR and RTL processing problem, you can either use replication definitions, disable the constraint checks on the tables that cause failure and produce the error messages, or you can use the **set dsi_command_convert to 'u2di'** clause with **alter connection** to convert **update** to **delete** followed by **insert**.

When you resume the DSI thread, Replication Server continues to use the original replication mode.

You can use **dsi_retry** with:

- **configure replication server** – to suspend replication at the server level for all connections.
- **alter connection** and **create connection** – to suspend replication at the connection level for the specified database.
- **create alternate connection** – to suspend replication for the specified replication path in a multipath environment.

For example, to suspend replication to the `iqdb` Sybase IQ replicate database in the IQSRVR data server when RTL fails to apply compilable transactions to `iqdb`, before you reenable RTL, enter:

```
alter connection to IQSRVR.iqdb
set dsi_retry to 1
go
```

The **dsi_retry** configuration at the connection level overrides the server-level configuration. You can use **dsi_retry** with any replicate database that supports HVAR, RTL, dynamic SQL, parallel DSI, or bulk copy-in.

See:

- *Tables with Referential Constraints* in the *Replication Server Administration Guide Volume 2* for HVAR, and in the *Replication Server Heterogeneous Replication Guide* for RTL.
- **dsi_command_convert** in *alter connection* in the *Replication Server Reference Manual*.

## Configuration Options and Example Error Messages for dsi_retry

Use **dsi_retry** to suspend replication immediately after any replication mode fails to apply transactions.

Set **dsi_retry** to:

- 0 – the default setting where Replication Server automatically switches to continuous replication mode when HVAR, RTL, parallel DSI, dynamic SQL or DSI bulk copy-in fails to apply transacations.
- 1 – to stop replication when HVAR or RTL fail to apply compilable transactions, or when parallel DSI, dynamic SQL, or DSI bulk copy-in fail to apply any transactions.
- 2 – to stop replication when the number of commands in a group containing failed transactions is smaller than the value of **dsi_compile_retry_threshold**, and HVAR or RTL fail to apply compilable transactions, or when parallel DSI, dynamic SQL, or DSI bulk copy-in fail to apply transactions.

After you set **dsi_retry** to the value that is relevant to the replication mode, Replication Server suspends replication when there is a failure in the replication mode.

### Example 1

An example of the message you see in the Replication Server log when you set **dsi_retry** to 1:

```
2012/09/18 23:06:35. A grouped transaction of 538 individual
transactions has failed in database 'GOME3_5551_IQ.iqrdb' with
'HVAR/RTL'.
DSI is now suspended because dsi_retry value is '1'
```

### Example 2

An example of the message you see in the Replication Server log when you set **dsi_retry** to 2:

```
2012/09/18 23:42:45. A grouped transaction of 20 individual
transactions has failed in database 'gome0_5551.tdb2'.
It will be broken into smaller groups and retried.

2012/09/18 23:42:46. Retry of a grouped transaction of 8 individual
transactions has failed in database 'gome0_5551.tdb2'.
It will be broken into smaller groups and retried.

2012/09/18 23:42:47. Retry of a grouped transaction of 4 individual
transactions has failed in database 'gome0_5551.tdb2'.
It will be broken into smaller groups.

2012/09/18 23:06:35. A grouped transaction of 538 individual
transactions has failed in database 'GOME3_5551_IQ.iqrdb' with
'HVAR/RTL'.
DSI is now suspended because dsi_retry value is '2'.
```

### Example 3

An example of the message you see in the Replication Server log when **dsi_retry** is set to 1 or 2, and dynamic SQL fails to apply transactions:

```
I. 2012/09/18 23:13:45. A grouped transaction of 20 individual
transactions has failed in database 'gome0_5551.tdb2' with 'DSQL/
BULK COPY'.
DSI is now suspended because dsi_retry value is '1'.
```

### Example 4

An example of the message you see in the Replication Server log when **dsi_retry** is set to 1 or 2, and DSI bulk copy fails to apply transactions:

```
I. 2012/09/18 23:13:45. A grouped transaction of 20 individual
transactions has failed in database 'gome0_5551.tdb2' with 'DSQL/
BULK COPY'.
DSI is now suspended because dsi_retry value is '1'.
```

### Example 5

An example of the message you see in the Replication Server log when **dsi_retry** is set to 1 or 2, and parallel DSI fails to apply transaction:

```
A parallel transaction has failed in database 'gome0_5551.tdb2'.
DSI is now suspended because dsi_retry value is '1'.
```

### Example 6

An example of the message you see in the Replication Server log when **dsi_retry** is set to 1 or 2, and replication with the continuous log-order language replication mode fails even after disabling the HVAR or RTL, dynamic SQL, and DSI bulk copy modes:

```
2012/09/18 23:18:05. A grouped transaction of 20 individual
transactions has failed in database 'gome0_5551.tdb2' with
'Language'.
DSI is now suspended because dsi_retry value is '1'.
```

# Common Error Messages

Learn about some of the most common Replication Server problems and the solutions to these problems.

To find an error description, search for:

- The error number of a Replication Server, **rs_init**, or Replication Agent error.
- The text of the **rs_init** error, or Replication Server informational or warning message.

Each error description includes:

- Symptom – includes the text of any error messages that are displayed in an error log. Also includes error conditions such as poor performance, replication failure, connection failure, and abnormal component termination.
- Explanation – describes the error messages and probable causes of the errors.
- Solution – describes procedures, workarounds, upgrades, and EBF information.

### See also

- *Troubleshooting Overview* on page 5
- *Error Messages and Error Logs* on page 7

## rs_init Error Messages

**rs_init** error messages do not have error numbers and are listed alphabetically.

### Cannot Find Entry for Adaptive Server

Adaptive Server entry does not exist in the `interfaces` file.

*Symptom*

This is reported in the **rs_init** error log:

```
2006/06/13 10:20:48 There is no entry for server
'westdss' in the interfaces file. The server must have
an existing entry in the interfaces file.
2006/06/13 10:20:48 Attribute 'rs_ds_name' could not be
set because'westdss' is an invalid value.
2006/06/13 10:20:48 Cannot execute configuration
because validation of input values failed.
2006/06/13 10:20:48 Exiting.
```

*Explanation*

When you execute **rs_init** with a resource file (**rs_init -r**), **rs_init** looks in the `interfaces` file for the Adaptive Server name that matches the one in the resource file.

The errors listed above occur because the Adaptive Server entry in the `interfaces` file does not match the resource file's Adaptive Server entry due to unprintable characters in the resource file.

*Solution*
Check for unprintable characters (such as control characters) in the Adaptive Server name entry in the resource file. The Adaptive Server name entry is specified as the *rs.rs_ds_name* parameter.

## Invalid Product Name

**rs_init** does not recognize the product parameter in the resource file.

*Symptom*
These messages are reported in the **rs_init** error log:

```
INTERNAL ERROR: Product '' not registered in internal registry.
Check the log (<filename>) for more information.  Please make a note
of this error and contact your Sybase representative.
In resource file '': '<resourcefilename>' is an invalid product name.
INTERNAL ERROR: Unable to load in resource file '<resourcefilename>'.
Check the log (<filename>) for more information.  Please make a note
of this error and contact your Sybase representative.
Exiting.
```

*Explanation*
This message will occur when you execute **rs_init** with a resource file (**rs_init -r** ) and one of the parameters in the resource file is missing the product name prior to the parameter. For instance, a line in the resource file similar to ".rs_rsdddb_size: 40" will generate this message because the full name of the parameter is "rs.rs_rsdddb_size".

*Solution*
Run **rs_init** again with the resource file and the complete parameter name.

## Unknown Host Machine Name

**rs_init** cannot find the Replication Server System Database (RSSD) machine's host name.

*Symptom*
These messages are reported in the **rs_init** error log:

```
2006/06/13 10:34:17 Running task: check the SQL Server.
2006/06/13 10:34:18 Unable to connect to SQL Server
'ost_agate_9'. Please make sure that the SQL Server is
running, and the SA password is correct.
2006/06/13 10:34:18 Task failed: check the SQL Server.
Terminating configuration.
2006/06/13 10:34:18 Configuration failed.
2006/06/13 10:34:18 Exiting.
```

*Explanation*
Sometimes directory services do not recognize the machine's host name and may instead recognize a machine only by its IP address.

*Solution*
Add a query line for the machine to the `interfaces` file and specify the host machine's IP address instead of its name. For example, for the `bss.bsslt.ch.corma.com` host name, substitute its IP address, `2.41.100.35`:

```
query tcp ether bss.bsslt.ch.corma.com 31440
query tcp ether 2.41.100.35 31440.
```

If you run **rs_init** again, you may see these messages in the **rs_init** error log, but you can ignore them:

```
Could not obtain Hostname for Internet address '2.41.100.35'.
Could not obtain Hostname for Internet address '2.41.100.41'.
Could not obtain Hostname for Internet address '2.41.100.35'.
Could not obtain Hostname for Internet address '2.41.100.41'. Running
task
to check the RSSD Adaptive Server.
```

# Replication Server Error Messages

Lists the Replication Server errors numerically and in ascending order. These error messages are recorded in the Replication Server error log.

## Error 21

Open Server call to routine **srv_spawn** failed.

*Symptom*
This message is reported after a new connection has started:

```
ERROR #21 DSI(126 U10PDTE.u10pdte) - (1426) Open server
call to routine 'srv_spawn' failed.
Open Server error: Error: 16115, State: 0, Severity 10
-- 'Could not start thread'.
```

*Explanation*
Replication Server ran out of Open Server threads. **srv_spawn** allocates these threads. The maximum number of Open Server threads that a Replication Server can use is equal to the value specified by the *num_threads* parameter.

*Solution*
Increase the number of Open Server threads that the Replication Server can use by increasing the value of the *num_threads* parameter. For example:

```
configure replication server
      set num_threads to '70'
```

## Error 1028

Error 1028 occurs when a data server returns an error to Replication Server. It is usually followed by several other errors.

### General Data Server Error

Troubleshooting procedure to general 1028 errors.

*Symptom*

Error 1028 appears in the Replication Server error log, but reports an Adaptive Server error:

```
E. 2005/06/12 15:25:44. ERROR #1028 DSI EXEC(107(2)
westernDS.westDB) - dsiqmint.c(3027) Message from
server: Message: ..., State ..., Severity... -- '...'.
```

*Explanation*

In some cases, the Data Server Interface (DSI) thread shuts down. Adaptive Server errors suspend the connection if:

- The Adaptive Server error is assigned the **retry_stop** or **stop_replication** error action.
- You executed the **suspend connection** command with the **nowait** clause.

If the DSI thread shuts down, you cannot simply resume the connection. Resuming the connection without fixing the problem sends the same transaction to the Adaptive Server and causes the same error.

*Solution*

Identify and fix the Adaptive Server problem that is causing the error. If you cannot correct the cause of the problem, you can, once you have tried all other solutions, resume the connection and skip the transaction. You can then manually apply the skipped transaction to the replicate table to resynchronize the primary and replicate tables. Skipping a transaction, however, causes inconsistencies between the primary and replicate databases, which you must manually fix in the replicate database.

#### See also

- *Implications of Skipping Transactions* on page 122

*Correcting Adaptive Server Error*

Identify and fix the Adaptive Server problem causing the replication error.

1. Log in to the RSSD and execute the **rs_helpexception**:

   ```
   1> rs_helpexception
   2> go
   ```

   ```
   Summary of Logged Transactions on 'westernRS'
   Total # of Logged Transactions = 1
   ```

```
Xact ID Org         Site Org User Org Date Dest Site #       Recs/
Xact
------------------- ------------- ------------- ------------- ---
------
107 mil01hprdss.eur eurian        Jun 13 2006   westernDS.eur
     3
For Detailed Information on a Logged Xact., type 'rs_helpexception
{XactID}'
(return status = 0)
```

2. To show the entire text of the transaction, execute **rs_helpexception** with the **v** option and the transaction ID from step 1:

```
1> rs_helpexception 107, v
2> go
```

You see:

```
Detailed Summary of Logged Transaction # 107 on 'westernRS'
Origin Site            Origin User   Org. Commit Date
#Cmds in Xact
--------------------   -----------   -----------------   --------
-----
westernDS.westDB      eurian       Jun 13 2006 12:24             3

Dest. Site             Dest. User    Date Logged
--------------------   -----------   -----------------
westernDS.westDB       …eurian       Jun 13 2006 12:27

This transaction was logged by the 'sysadmin log_first_tran'
command.

Rejected Records
textval
-------------------------------------------------------------------
A0100distribute :origin_time='Jun 13
12:24:24:416PM',:origin_user='',
:mode=1
begin transaction 'logexec' for 'eurian'/'******'
begin transaction
A0100distribute :origin_user='',:mode=1
exec "TT"."so_req_rep_all_allcon" @"p01"=80000709,@"p02"='MIL'
execute tt_act_rep_all_allcon @p01 = 80000709, @p02 = 'MIL'
A0100distribute :origin_time='Jun 13
12:24:416PM',:origin_user='',
:mode=1
commit transaction
execute rs_update_lastcommit @origin = 107,@origin_qid
=0x00000001004620
d300019296000effffffff000000008910009bd7cd0001000000000001,
@ secondary_qid
=0x00000000000000000000000000000000000000000000000000000000
00000000000000000000, @origin_time = 'Jun 13 12:27:227PM'
commit transaction
(return status = 0)
```

This text corresponds to what is sent to the database (in this case, function strings have been applied).

3. Use the transaction information to manually apply the update to the replicate database.
4. After fixing the error in the database, resume the connection in the Replication Server:

```
> resume connection to westernDS.westDB
     skip transaction
```

5. Delete the transaction from the exceptions log to keep the log small:

```
1> exec rs_delexception 107
2> go
```

You can use **rs_delexception_id** to delete a range of transactions identified by transaction ID. You can also use **rs_delexception_date** to delete a range of transactions identified by transaction date, and **rs_delexception_range** to delete a range of transactions identified by originating site or user, or destination site. See the descriptions of the stored procedures in *Replication Server Reference Manual > RSSD Stored Procedures* for complete usage information and more examples.

## DSI Shuts Down Because of SQL Error in Transaction

Occurs when a Replication Server transaction that is sent to the replicate Adaptive Server has a SQL error. An Adaptive Server error is included in the text of the Replication Server error.

### *Symptom*

The following error messages are reported in the Replication Server error log:

```
E. 2006/06/13 12:31:29. ERROR #1028 DSI(western.west1)-
dsiqmint.c(3645) Message from server: Message: 229,
State: 1, Severity: 14-- 'DELETE permission denied on
object real_run, database west1, owner dbo'.
H. 2006/06/13 12:31:29. THREAD FATAL ERROR #5049
DSI(western.west1) - dsiqmint.c(3652) The DSI thread
for database 'western.west1' is being shutdown. DSI
received data server error #229 which is mapped to
STOP_REPLICATION. See logged data server errors for more
information. The data server error was caused by RS
output command #1 mapped from input command #2 of the
failed transaction.
I. 2006/06/13 12:31:29. The DSI thread for database
'western.west1' is shutdown. The Message from server:
text is the message you would get from the Adaptive
Server had you sent the command batch to the server with
isql. The text is taken from the table sysmessages. In
isql you would get: Msg 229, Level 14, State 1: DELETE
permission denied on object real_run, database west1,
owner dbo
```

### *Explanation*

These errors occur when a Replication Server transaction that is sent to the replicate Adaptive Server includes a SQL error. The replicate Adaptive Server detects the SQL error in the transaction and returns a message to the Replication Server.

The Replication Server responds by shutting down the connection and suspending replication; this prevents data inconsistencies between the primary and replicate sites. It allows the user to correct the problem at the replicate Adaptive Server and to maintain data consistency.

For example, when a SQL semantic error occurs in a referential integrity or duplicate keys situation, Adaptive Server sends its message back to Replication Server. In response, Replication Server shuts down the connection and records the Replication Server error in the Replication Server error log. The Adaptive Server error is referenced in the body of the Replication Server error.

### *Solution*

Fix the problem in Adaptive Server and resume the connection as described in *Replication Server Troubleshooting Guide > Common Errors > 1028 > General Data Server Error*. If the problem persists, verify that **autocorrection** for the subscription has been turned on in the `rs_repobjs.attributes` table of the Replication Server System Database (RSSD). Also, check for triggers that enforce referential integrity rules.

### See also

- *General Data Server Error* on page 38

## Adaptive Server and DB2 Table Names Do Not Match

Occurs when a DB2 table with an uppercase name is sent to an Adaptive Server.

### *Symptom*

These messages are reported in the Replication Server error log:

```
E. 2006/06/13 12:31:29. ERROR #1028 DSI(DSMA1.apptst) -
dsiqmint.c(3668)
Message from server: Message: 208, State: 1, Severity: 16 -- 'TEST
not found.
Specify owner.objectname or use sp_help to check whether the object
exists
(sp_help may produce lots of output). '.
H. 2006/06/13 12:31:29. THREAD FATAL ERROR #5049 DSI(DSMA1.apptst) -
dsiqmint.c(3675) The DSI thread for database 'DSMA1.apptst' is being
shutdown. DSI received data server error #208 which is mapped to
STOP_REPLICATION.
See logged data server errors for more information. The data server
error
was caused by RS output command #1 mapped from input command #2 of
the failed
transaction.
```

```
E. 2006/06/13 12:31:29. ERROR #32032 LTM USER(TCPIP.ZD60) - /nrm/
nrm.c(1658)
No table with name 'TEST' is defined at repserver with id TCPIP.ZD6
```

### *Explanation*

DB2 object names are in uppercase while Adaptive Server object names can be in mixed case.

*Solution*
Any of:

- Replicate into VIEWS with uppercase object names.
- Create a table name in uppercase, and use custom function strings with column names that match the case of the columns in the replicate table.
- Create the Adaptive Server object names in uppercase to match the DB2 object names.
- If the Adaptive Server table uses lowercase names, use function strings to force the Replication Server to generate lowercase table and column names when transactions are sent to the Adaptive Server.

### Adaptive Server last-chance Threshold Passed

Occurs when the last-chance threshold is reached. This error includes references to the Adaptive Server error 7415.

*Symptom*
These errors are recorded in the Replication Server error log:

```
I. 2006/06/13 10:45:07. Message from server: Message: 7415, State: 1,
Severity: 10 - - 'The transaction log in database northDB is almost
full.
Your transaction is being suspended until space is made available in
the
log.'.
E. 2006/06/13 10:45:07. ERROR #5046 DSI(axp st.northDB) - /
dsioqid.c(1638)
When executing the rs_get_lastcommit function in database 'axp
st.northDB',
received data server errors. See logged data server errors for more
information.
```

*Explanation*
By default, the replicate Adaptive Server suspends all transaction processing when the destination database log segment size becomes greater than the Adaptive Server last-chance threshold. The last-chance threshold is a parameter that keeps the log from expanding to greater than the maximum size of the log.

This problem is related to Data Server Interface (DSI) shutting down because the replicate database log is full. If the database is the Replication Server System Database (RSSD), other serious consequences can occur.

*Solution*
Reduce the size of the log by dumping or truncating it:

1. Manually dump the log using these commands in the RSSD:

   ```
   > sp_helpdb northDB
   > go
   > dump tran northDB to ...
   ```

```
> go
> sp_helpdb northDB
> go
```

where `northDB` is the replicate database with a full log.

2. If this step fails, truncate the log by executing **dump tran** with the **truncate_only** or **no_log** option.

### See also

- *RSSD Log Device Full* on page 49

## DSI Shuts Down Because Replicate Database Log is Full

The Data Server Interface (DSI) thread to the replicate data server has shut down because the Adaptive Server log for the database is full.

### *Symptom*

The DSI thread shuts down and this message is reported in the Replication Server error log:

```
E. 2006/06/13 10:49:07. ERROR #1028 DSI EXEC(107(1) SYDNEY_DS.pubs2)
-
dsiqmint.c( 2361) Message from server: Message: 1105, State 3,
Severity 17
-- 'Can't allocate space for object 'syslogs' in database 'pubs2'
because
the 'logsegment' segment is full. If you ran out of space in syslogs,
dump
the transaction log. Otherwise, use ALTER DATABASE or
sp_extendsegment to
increase the size of the segment.
H. 2006/06/13 10:49:07. THREAD FATAL ERROR #5049 DSI EXEC(107(1)
SYDNEY_DS.pubs2) - dsiqmint.c(2368) The DSI thread for database
'SYDNEY_DS.pubs2' is being shutdown.
DSI received data server error #1105 which is mapped to
STOP_REPLICATION.
See logged data server errors for more information. The data server
error
was caused by output command #1 mapped from input command #1 of the
failed
transaction.
```

### *Explanation*

A full Adaptive Server log may be caused by problems at the replicate database or the replicate Replication Server.

This problem is related to the last-chance threshold being reached, which occurs when the Adaptive Server log for the database is almost full.

*Solution*

1. Check if there is enough space in the stable device to increase the size of the Adaptive Server log, and add space to the stable device if needed.
2. Add space to the Adaptive Server log.
3. Make sure the DSI thread to the data server and the RSI thread to the Replication Server:
   - Start
   - Resume a connection
   - Replicate new transactions after a closed connection

### See also

- *Adaptive Server last-chance Threshold Passed* on page 42

## Accessing a Database While in Recovery

Occurs when a Replication Server attempts to use an Adaptive Server database that is in the process of recovery.

*Symptom*

These messages are recorded in the Replication Server error log:

```
E. 2006/06/13 10:53:36. ERROR #1028 DSI(western.west1) - /
dsiexec.c(306)
Message from server: Message: 921, State: 1, Severity: 14 --
'Database
'west1' has not been recovered yet - please wait and try again.'.
I. 2006/06/13 10:53:36. Message from server: Message: 5701, State: 1,
Severity: 10 - - 'Changed database context to 'master'.'.
E. 2006/06/13 10:53:36. ERROR #5051 DSI(western.west1) - /
dsiexec.c(314)
Received errors from database 'western.west1'. See logged ct-lib and
data
server messages for more information.
```

*Explanation*

A Replication Server cannot connect to a recovering Adaptive Server database. Replication Server automatically attempts to connect to databases (to which connections have been created) when it:

- Starts
- Resumes a connection
- Replicates new transactions after a closed connection

*Solution*

The connection resumes automatically after the database has recovered.

### Replication Server Not Found

Sybase IQ failed to connect to the replicate Replication Server.

*Symptom*

The Data Server Interface (DSI) shuts down and SQL Anywhere® Error -1003002 is recorded in the Replication Server error log:

```
E. 2010/09/20 16:24:33. ERROR #1028 DSI EXEC(103(1) mrpserver.mrp) -
dsiqmint.c(4218)
Message from server: Message: -1003002, State 0, Severity 14 -- 'SQL
Anywhere Error -1003002:
CtLibrary Error: 3, Severity: 0, Origin: 8, Layer: 6
Error Message: ct_connect(): directory service layer: internal
directory control layer error:
Requested server name not found.
OS Error: 0, OS Message:
```

*Explanation*

Replication to Sybase IQ using real-time loading (RTL) replication has not been properly configured. Sybase IQ must be able to connect to Replication Server and retrieve data each time Replication Server sends an **INSERT ... LOCATION** statement to Sybase IQ.

*Solution*

Create an entry for the replicate Replication Server in the `interfaces` file of the replicate Sybase IQ server. This allows Sybase IQ to connect to Replication Server using the user name and password specified in the database connection. The maintenance user must be a valid user in Replication Server with system administrator privileges and whose password matches the password that Replication Server uses to log in to Sybase IQ.

### Sybase IQ Invalid Permission in a Multiplex Environment.

Occurs when connection to coordinator node is not happening. Because in an IQ Multiplexed system, the coordinator node is the only node that can do the **lock table** command.

*Symptom*

These messages are reported in the Replication Server error log:

```
E. 2010/09/14 08:51:13. ERROR #1028 DSI EXEC(104(1) pocmpx.iqdb) -
dsiqmint.c(4234) Message from server: Message: -1004015, State 0,
Severity 14 -- 'SQL
Anywhere Error -1004015: Permission denied: Command not allowed on
Multiplex Writer servers. (saint_iqthresholdddl.cxx 14936)'.
```

*Explanation*

Because Replication Server connects and issues the **lock table** command, the connection from the Replication Server to Sybase IQ in a Multiplex environment must be made to the coordinator node.

*Solution*
Change the interfaces file entry for IQ that the Replication Server uses, to connect to the coordinating node.

See the *Replication Server Heterogeneous Guide > Replicating into Sybase IQ*.

## Error 5095

The column length returned by the rs_get_lastcommit function is incorrect.

*Symptom*

```
E. 2011/11/23 13:44:15. ERROR #5095 DSI(138 spotak1520iq.iqdemo) - /
dsioqid.c(1786)
       The second and third columns returned by the rs_get_lastcommit
function should have length 72. The one returned from database
'spotak1520iq.iqdemo'    for column 2 is 74.
I. 2011/11/23 13:44:15. The DSI thread for database
'spotak1520iq.iqdemo' is shutdown.
```

*Explanation*
This problem occurs when replicating to Sybase IQ and the value of the configuration property parameter, ASE_BINARY_DISPLAY, for maintenance user is set to on.

*Solution*
Set ASE_BINARY_DISPLAY to off, which is the default value for a Sybase IQ server.

## Error 7035

The Replication Server is out of memory.

*Symptom*
After a connection is started, these messages are reported in the Replication Server error log:

```
I. 2006/06/13 10:58:42. The DSI thread for database
‘westernDS.westDB’ is
started.
E. 2006/06/13 10:58:42. ERROR #7035 DIST(westernDS.westDB) - m/
memseg.c(771)
.........Additional allocation would exceed the memory_limit of
‘20000’
specified in the configuration.
Increase the value of the memory_limit parameter in the rs_config
table and
restart the Repserver.
```

A second attempt at resuming the connection shows only this message:

```
I. 2006/06/13 11:08:06. Attempt to start a DSI thread
for database ‘westernDS.westDB’ that has already been
started.
```

Running **admin who_is_down** at the Replication Server indicates that threads are down.

This example indicates that the DIST and DIST EXEC threads are down:

```
Spid   Name         State   Info
----   ---------    -----   -------------------------
       DIST         Down    westernDS.westDB
       DIST EXEC    Down    105:1 westernDS.westDB
```

*Explanation*

Replication Server ran out of segment memory. The maximum amount of operating system memory that a Replication Server can use is equal to the value specified by the *memory_limit* parameter. Replication Server directly uses operating system memory.

*Solution*

1. Increase the value of the *memory_limit* parameter.
2. Restart the Replication Server.

## Error 8039

The amount of memory available to the Distributor for messages waiting to be written to the outbound queue is being reached.

*Symptom*

This error message is reported in the Replication Server error log:

```
E. 2008/05/21 08:37:50. ERROR #8039 SQM(16877328:0 TRADE_REP) - tr/
mdext.c(2009)
MD failed to wake someone waiting for memory from
source=TRADEDS.tradedb.
```

*Explanation*

This is an informational message indicating that the maximum value for **md_sqm_write_request_limit** parameter is being reached.

*Solution*

If reported frequently, increase the maximum value for **md_sqm_write_request_limit** parameter.

## Error 8040

Stable Queue Manager (SQM) attempted to awaken a Distributor thread that is actually up.

*Symptom*

This error message is reported in the Replication Server error log:

```
E. 2011/03/28 06:27:44. ERROR #8040 SQM(163:0 TRADEDS.tradedb) - tr/
mdext.c(2066)
MD failed to wake someone waiting for flush from source=TRADE_REP.
```

*Explanation*
Indicates that the SQM Writer has attempted to awaken the Distributor, but the Distributor was not asleep. If the Distributor threads are up and replication is proceeding normally, this message can be considered informational.

*Solution*
If the Distributor threads are down, resume them.

## Error 11061

Replication Server System Database (RSSD) problems.

### RSSD Deadlocks

Replication Server System Database (RSSD) deadlocks usually occur when commands for the RSSD are issued faster than the server can process them. Deadlocks may occur even on a fast machine and network when you run scripts that create, alter, or delete many subscriptions or replication objects.

*Symptom*
The RSSD stops responding and you see these messages in the Replication Server error log:

```
E. 2006/06/13 11:14:12. ERROR #11061 USER(rho_dbo) - s/
stscol.c(1717) Check
the log for error messages from RSSD.
E. 2006/06/13 11:18:22. ERROR #1028 USER(rho_dbo) - s/stscol.c(1717)
Message
from server: Message: 1205, State: 2, Severity: 13 -- 'Your server
command
(process id #14) was deadlocked with another process and has been
chosen as
deadlock victim. Re-run your command.'.
```

*Explanation*
RSSD deadlocks may occur when you:

- Create routes in parallel within a star configuration. A star configuration has one primary Replication Server with only direct routes to other destination Replication Servers, and each destination Replication Server has only one direct route back to the primary Replication Server.
- Create, activate, or validate subscriptions in one or more Replication Servers.
- Drop replication definitions in parallel in different Replication Servers.

**Note:** In a production environment, deadlock situations on the replicate database are automatically handled by the Replication Server.

*Solution*
If routes are deadlocked, drop the routes and re-create them sequentially, allowing one minute between each creation.

If an RSSD deadlock occurs during the activation or validation of subscriptions:

1. Use **rs_helpsub** in the RSSD or **check subscription** in the Replication Server to check for subscriptions with status “Active/Activating” instead of “Active/Unknown.”
2. Use the **without purge** option to drop the “Active/Activating” subscriptions, then recreate the subcriptions.

If deadlocks occur while you are dropping subscriptions, drop them again.

To prevent a large number of deadlocks, do not simultaneously load several scripts into Replication Server. In extreme situations, avoid loading scripts simultaneously in different Replication Servers; instead run scripts sequentially.

### RSSD Log Device Full

The Replication Server System Database (RSSD) log space has fallen critically low.

#### *Symptom*

These messages are reported in the Replication Server error log:

```
E. 2006/06/13 10:35:15. ERROR #11061 USER(western_dbo)
- s/stscol.c(1717) Check the log for error messages from
RSSD.
I. 2006/06/13 10:35:15. Message from server: Message:
7412, State: 1, Severity: 10 - - ‘Space available in the
log segment has fallen critically low in database
‘rssd’.
All future modifications to this database will be
suspended until the log is successfully dumped and space
becomes available.’.
I. 2006/06/13 10:35:15. Message from server: Message:
7415, State: 1, Severity: 10 - - ‘The transaction log
in database rssd is almost full.
Your transaction is being suspended until space is made
available in the log.’.
```

#### *Explanation*

During subscription validation, the RSSD ran out of log space even though the **truncate on checkpoint** option was set. Replication Server halted and Adaptive Server suspended modifying all transactions. After you resolved the log space issue, Replication Server continued to process the subscription but did not validate it correctly.

#### *Solution*

Reduce the size of the log by dumping or truncating it:

1. Manually dump the log using these commands in the RSSD:

   ```
   > sp_helpdb RSSD
    > go
    > dump tran RSSD to ...
    > go
   ```

```
> sp_helpdb RSSD
> go
```

2. If this step fails, truncate the log by executing **dump tran** with the **truncate_only** or **no_log** option.

To prevent this error, monitor the RSSD log. If the RSSD log becomes more than 80% full:

1. Suspend the operating system process that creates replicated objects (subscriptions).
2. Wait one minute to allow Replication Server to finish its transactions.
3. Dump the transaction log.
4. Resume the process.

## Error 13045

Replication has been suspended because Replication Server System Database (RSSD) restarted.

*Symptom*

These messages are reported in the Replication Server error log:

```
E. 2006/06/13 14:50:16. ERROR #13045 SQT(101:1 DIST westss.eastlp) -
seful/cm.c(3914)
Failed to connect to server 'westss' as user 'westrs_rssd_prim'. See
CT-Lib
and/or server error messages for more information.
I. 2006/06/13 14:50:17. Trying to connect to server 'westss' as user
'westrs_rssd_prim' ......
```

After the Adaptive Server with the RSSD has restarted, these messages are reported in the Replication Server error log:

```
E. 2006/06/13 17:04:52. ERROR #1027 dSUB( ) -
seful/cm.c(3909)
Open Client Client-Library error: Error: 84083972,
Severity 5 -- 'ct_connect():
network packet layer: internal net library error: Net-
Lib protocol driver call to connect two endpoints
failed', Operating System error 0 -- 'Socket connect
failed - errno 146 Connection refused'.
E. 2006/06/13 17:04:52. ERROR #13045 dSUB( ) -
seful/cm.c(3914)
Failed to connect to server 'westss' as user 'amerttp'.
See CT-Lib and/or server error messages for more
information.
I. 2006/06/13 17:04:52. Trying to connect to server
'westss' as user 'westrs_rssd_prim' ......
E. 2006/06/13 17:04:57. ERROR #1027 dSUB( ) -
seful/cm.c(3909)
Open Client Client-Library error: Error: 84083972,
Severity 5 -- 'ct_connect():
network packet layer: internal net library error: Net-
Lib protocol driver call to connect two endpoints
failed', Operating System error 0 -- 'Socket
connectfailed - errno 146 Connection refused'.
```

```
E. 2006/06/13 17:05:56. ERROR #13043 USER(westss_ra) - ul/
cmapp.c(888)
Failed to execute the 'USE westss_rssd' command on
server 'westss'. See CT-Lib and SQL Server error
messages for more information.
E. 2006/06/13 17:05:56. ERROR #1028 USER(westss_ra) -
ul/cmapp.c(888)
Message from server: Message: 911, State 2, Severity 11
-- 'Attempt to locate entry in sysdatabases for database
'westss_rssd' by name failed - no entry found under that
name. Make sure that name is entered properly.'.
I. 2006/06/13 17:05:56. Message from server: Message:
5701, State 1, Severity 10 -- 'Changed database context to
'master'.'.
E. 2006/06/13 17:05:56. ERROR #13045 USER(westss_ra) - seful/
cm.c(3318)
Failed to connect to server 'westss' as user
'westrs_rssd_prim'. See CT-Lib and/or server error
messages for more information.
E. 2006/06/13 17:05:56. ERROR #1028 USER(westss_ra) -
seful/cm.c(3318)
Message from server: Message: 911, State 2, Severity 11
-- 'Attempt to locate entry in sysdatabases for database
'westss_rssd' by name failed - no entry found under that
name. Make sure that name is entered properly.'.
I. 2006/06/13 17:05:56. Message from server: Message:
5701, State 1, Severity 10
-- 'Changed database context to 'master'.'.
E. 2006/06/13 17:05:56. ERROR #13043 dREC(dREC)--
ul/cmapp.c(888)
Failed to execute the 'USE westss_rssd' command on
server 'westss'. See CT-Lib and SQL Server error
messages for more information.
```

### *Explanation*

The Adaptive Server that controls the Replication Server System Database (RSSD) was shut down and restarted while the Replication Server was running. The Distributor (DIST) and Stable Queue Transaction (SQT) threads to the databases controlled by the Replication Server were terminated. Replication to those databases was terminated and does not resume even after the RSSD becomes available again.

Running the **admin who_is_down** command at the Replication Server shows that both DIST and SQT threads are down:

```
Spid    Name      State      Info
----    ------    -------    ----------------------
        DIST      Down       westernDS.westDB
        SQT       Down       105:1 westernDS.westDB
```

### *Solution*

1. At the Replication Server, execute **resume distributor** for each database to resume SQT and DIST threads.

2. Run **admin who_is_down** at each database to verify that the SQT and DIST threads are up.

## Error 15020

Replication definition cannot be found.

*Symptom*

These messages are reported in the Replication Server error log:

```
Error #15020: "'%s' doesn't exist."
```

*Explanation*

The replication definition for which you want to create a subscription cannot be found.

*Solution*

- Verify that a replication definition has been created for the primary version of the table that is to be replicated.
- If the primary Replication Server is not the same as the replicate Replication Server, verify that:
  - A route has been created from the primary Replication Server to the replicate Replication Server, or
  - The replication definition has been replicated.
- Verify that subscription data definition language (DDL) commands were entered at the replicate Replication Server.

## Error 15040

Connection to the replicate database does not exist.

*Symptom*

This message is reported in the Replication Server error log:

```
Error #15040: "This database '%s.%s' is not controlled
by this site."
```

*Explanation*

A connection from the replicate Replication Server to the database has not been created.

*Solution*

Use the **create connection** command at the replicate Replication Server to make a connection to the replicate database. To create a connection to an Adaptive Server replicate database, use **rs_init**.

## Error 15052

The primary and replicate databases are the same.

*Symptom*

This message is reported in the Replication Server error log:

```
Error #15052: "The replicate database '%s.%s' for
subscription '%s' is the same as the primary database
for the replication definition '%s'. This is not
allowed."
```

*Explanation*

You cannot create a subscription in which the replicate database is the same as the primary database.

*Solution*

1. Create the replicate table on a different database.
2. Execute the **create subscription** command again.

## Error 28028

The **connect source** command has not been executed.

*Symptom*

This message is reported in the Replication Server error log:

```
Error #28028: "Connect source as user '%s' failed at PRS
'%s' for subscription '%s', for replication definition
'%RS_RSID' with replicate at '%RS_SITEID'."
```

*Explanation*

The replicate Replication Server failed to execute the **connect source** command at the primary Replication Server.

*Solution*

The subscription recovery daemon should recover the subscription.

## Error 29024

Replication Server cannot find a matching function string for a function.

*Symptom*

The message reported in the Replication Server error log is similar to:

```
Jan 26 11:27:59.300 2006: Error: 29204, Line: 864, File:
'generic/dsi/fstrmap.c' - Cannot find a matching
function string for function 'stocks.rs_insert' and
function class 'rs_sqlserver_function_class'.
```

*Explanation*

This error occurs when Replication Server cannot find a function string to match a function. The Data Server Interface (DSI) thread is suspended.

*Solution*

1. Query the Replication Server System Database (RSSD) at the primary site of the function string to determine if the function string exists there. Use this query for strings with replication definition scope:

   ```
   select name from rs_funcstrings, rs_functions,
      rs_classes, rs_objects
   where rs_funcstrings.classid = rs_classes.classid
      and rs_funcstrings.funcid = rs_functions.funcid
      and classname = function_class_name
      and classtype = 'F'
      and rs_functions.objid = rs_objects.objid
      and rs_objects.objname = replication_definition
      and objtype = 'R' and funcname = function_name
   ```

   If the function string does not exist at the primary site, continue with step 2.

   If the function string does exist at the primary site, go to step 3.
2. If the function string does not exist at the primary site, create it there. If you omit the **output** clause from the string, Replication Server generates the default function string.
3. If the function string does exist at the primary site, it has not replicated to the Replication Server at the replicate site. Verify that the RSSD at the primary site is being replicated to the replicate site. If changes at the primary site are being replicated to the replicate site, then you may need to wait for the function strings to arrive. Then restart the DSI.
4. For **rs_select** and **rs_select_with_lock**, there must be a function string for which the input template matches the **where** clause of the **select** command. Extend the above query with a join to `rs_systext` to retrieve the input templates of function strings.

   If the DSI receiving the error is communicating with a server using a function-string class other than the `rs_sqlserver_function_class`, `rs_default_function_class`, or `rs_db2_function_class`, you may still need to create the function string.

### See also

- *Troubleshooting Replication Failures* on page 21

## Error 37022

Permission to create a subscription is required.

*Symptom*

This message is reported in the Replication Server error log:

```
Error #37022: "PRIMARY SUBCRIBE permission is required
to execute the subscription command."
```

*Explanation*
You do not have permission to create a subscription. You must have **primary subscribe**, **create object**, or **sa** permission at the primary Replication Server.

*Solution*
Use the **grant** command to change the permissions.

### Error 37023

Permission to create an object is required.

*Symptom*
This message is reported in the Replication Server error log:

```
Error #37023: "CREATE OBJECT permission is required to
execute command."
```

*Explanation*
You do not have the required permission.

*Solution*
Use the **grant** command to change the permission to **create object**.

## Replication Server Informational and Warning Messages

Informational and warning messages are reported in the Replication Server error log.

### Cached Row for System Table Was Swapped Out

A cached row is swapped out to accommodate another row.

*Symptom*
This informational message appears many times in the Replication Server error log:

```
I. 2006/06/13 15:39:53. A cached row for system table
'rs_columns' was swapped out of the cache in order to
accomodate another row.
```

*Explanation*
When there is no space available in the cache for any more rows in a specific Replication Server System Database (RSSD) system table, that system table's oldest rows are swapped out of the cache.

The *sts_cachesize* parameter specifies the maximum number of rows that are cached for a single RSSD system table.

*Solution*
Increase the value of the *sts_cachesize* parameter for the RSSD.

## Detecting Loss for Database

Usually occurs after a **rebuild queues** command is executed. This command is typically issued after a stable device failure.

*Symptom*
After you have issued the **rebuild queues** command and the data loss detection process has completed, see the Loss Status column in the output from **admin health** and **admin who, sqm** to check for data loss in the queues, or data loss detection messages in the Replication Server error log. Perform data loss detection only when there are primary transactions to be replicated. The Replication Monitoring Services (RMS) heartbeat feature replicates a primary transaction to the destination data server. You can activate the heartbeat feature for the connection to force data loss detection to be performed.

**Note:** Sometimes these messages do not appear at the end of the error log file, so you must search back through the log for them. If you restart the Replication Server, these messages appear again.

```
I. 2006/06/13 15:48:32. Rebuild Queues: Starting
I. 2006/06/13 15:48:33. Disconnecting Replication Agent
for westss.westrs_rssd. Replication Agent will shutdown
I. 2006/06/13 15:48:33. Disconnecting Replication Agent
for westss.westrs_rssd to Rebuild
I. 2006/06/13 15:48:33. Resetting Replication Agent
starting log position for westss.westrs_rssd
I. 2006/06/13 15:48:33. Resetting Replication Agent
starting log position for westss.eastlp
I. 2006/06/13 15:48:33. Shutting down distributor for
101.
I. 2006/06/13 15:48:33. A request to shutdown/suspend
the distributor for 101 has been received.
I. 2006/06/13 15:48:33. The distributor for
'westss.westrs_rssd' is shutting down
I. 2006/06/13 15:48:33. Shutting down distributor for
102.
I. 2006/06/13 15:48:33. A request to shutdown/suspend
the distributor for 102 has been received.
I. 2006/06/13 15:48:34. The distributor for
'westss.eastlp' is shutting down
I. 2006/06/13 15:48:34. Shutting down the DSI thread for
'westss.westrs_rssd'.
I. 2006/06/13 15:48:34. The DSI thread for database
'westss.westrs_rssd' is shutdown.
I. 2006/06/13 15:48:34. DSI: enabled loss detection for
'westss.westrs_rssd'.
I. 2006/06/13 15:48:34. Shutting down the DSI thread for
'westss.eastlp'.
```

```
I. 2006/06/13 15:48:34. The DSI thread for database
'westss.eastlp' is shutdown.
```

```
I. 2006/06/13 15:48:34. DSI: enabled loss detection for
'westss.eastlp'.
I. 2006/06/13 15:48:34. Shutting down the DSI thread for
'westss.westlp'.
I. 2006/06/13 15:48:35. The DSI thread for database
'westss.westlp' is shutdown.
I. 2006/06/13 15:48:35. DSI: enabled loss detection for
'westss.westlp'.
I. 2006/06/13 15:48:35. Rebuild queues: deleting queue
103:0
I. 2006/06/13 15:48:35. SQM stopping: 103:0
westss.westlp
I. 2006/06/13 15:48:35. Rebuild queues: done rebuilding
queue 103:0. Restarting.
I. 2006/06/13 15:48:35. Rebuild queues: deleting queue
102:1
I. 2006/06/13 15:48:35. SQM stopping: 102:1
westss.eastlp
I. 2006/06/13 15:48:35. SQM starting: 103:0
westss.westlp
I. 2006/06/13 15:48:35. Rebuild queues: done rebuilding
queue 102:1. Restarting.
I. 2006/06/13 15:48:35. Rebuild queues: deleting queue
102:0
I. 2006/06/13 15:48:36. SQM stopping: 102:0
westss.eastlp
I. 2006/06/13 15:48:36. SQM starting: 102:1
westss.eastlp
I. 2006/06/13 15:48:36. Rebuild queues: done rebuilding
queue 102:0. Restarting.
I. 2006/06/13 15:48:36. Rebuild queues: deleting queue
101:1
I. 2006/06/13 15:48:36. SQM stopping: 101:1
westss.westrs_rssd
I. 2006/06/13 15:48:36. SQM starting: 102:0
westss.eastlp
I. 2006/06/13 15:48:36. Rebuild queues: done rebuilding
queue 101:1. Restarting.
I. 2006/06/13 15:48:36. Rebuild queues: deleting queue
101:0
I. 2006/06/13 15:48:36. SQM stopping: 101:0
westss.westrs_rssd
I. 2006/06/13 15:48:36. SQM starting: 101:1
westss.westrs_rssd
I. 2006/06/13 15:48:37. Rebuild queues: done rebuilding
queue 101:0. Restarting.
I. 2006/06/13 15:48:37. SQM starting: 101:0
westss.westrs_rssd
I. 2006/06/13 15:48:37. Starting DIST for 101:1.
I. 2006/06/13 15:48:37. Starting DIST for 102:1.
I. 2006/06/13 15:48:37. DIST for 'westss.westrs_rssd'
is Starting
I. 2006/06/13 15:48:37. DIST for 'westss.pdb' is
```

```
Starting
I. 2006/06/13 15:48:37. Starting the DSI thread for
'westss.westrs_rssd'.
I. 2006/06/13 15:48:38. Starting the DSI thread for
'westss.westlp'.
I. 2006/06/13 15:48:38. The DSI thread for database
'westss.westrs_rssd' is started.
I. 2006/06/13 15:48:39. Starting the DSI thread for
'westss.eastlp'.
I. 2006/06/13 15:48:40. The DSI thread for database
'westss.eastlp' is started.
I. 2006/06/13 15:48:41. The DSI thread for database
'westss.eastlp' is started.
I. 2006/06/13 15:48:41. Rebuild Queues: Complete
I. 2006/06/13 15:48:44. DSI: detecting loss for database
'westss.eastlp' from origin 'westss.westlp' date =
'Jun 13 2006  2:36:49:783PM', qid=0000000000001d
240000054b00090000054b0007000097df00f0d41700000000000
0001.
```

*Explanation*

Replication Server detected loss for a primary Replication Server or database. User transactions from the primary Replication Server or database are no longer accepted. If the replication system has more than one Replication Server, then the detected data loss may be either a Data Server Interface (DSI) loss or an Replication Server Interface (RSI) loss. If the replication system has only one Replication Server, the detected data loss is a DSI loss. A DSI loss means that data was not replicated from the Replication Server to the replicate database. An RSI loss means that data was not replicated from one Replication Server to another Replication Server.

*Solution*

For information on correcting data losses, see *Replication Server Administration Guide Volume 2*.

To prevent data loss, use DSI or RSI save intervals to create a backup strategy with logs large enough to keep all messages.

## DSI Detected rs_update_lastcommit Not Marked as Replicated

The **rs_update_last commit** stored procedure is not marked as replicated.

*Symptom*

A warm standby database disconnects and these error are reported in the Replication Server error log:

```
I. 2006/06/13 15:39:53. DSI for %RS_SITEID detected that
stored procedure rs_update_lastcommit is not marked as
replicated. Please execute sp_setreplicate and resume
connection
```

```
When active DSI comes up in presence of the Warm Standby
it checks whether rs_update_lastcommit stored procedure
is marked as  replicated. This DSI detected a problem
with replication status of  this stored procedure.
Please, correct the problem and resume connection
```

*Explanation*

This error occurs during a warm standby installation. You used **isql** to execute the **create connection** command and did not mark **rs_update_lastcommit** as replicated.

*Solution*

1. Set replication status on **rs_update_lastcommit** by marking it using **sp_setrepproc**.
2. Resume connection.

Instead of using the **create connection** command, you can use **rs_init** to create a connection to a warm standby database. This message does not occur when you use **rs_init** to set up the warm standby.

When you use **rs_init** to set up your active and standby database, the **rs_update_lastcommit** stored procedure is automatically marked for replication.

## Stable Storage Use is Above 75 Percent

The total size of all the queues on a Replication Server has passed the set threshold.

*Symptom*

This message appears in the Replication Server error log:

```
W. 2006/06/13 18:41:12. WARNING #6089 SQM(129:0
TTTdss.eanp) - qm/sqmsp.c(1317) WARNING: Stable Storage
Use is Above 75 percent
```

*Explanation*

A Replication Server has two default thresholds (75% and 90%) for stable devices. A warning message is issued each time these thresholds are exceeded.

*Solution*

When the 75% and 90% thresholds have been exceeded:

1. Verify that the replicate database is running.
2. If all other components are functional and the inbound queues are growing, shut down the corresponding Adaptive Server and check for an open transaction.
3. If the outbound queues are growing, check the DSI save interval with low block usage.

# Connector Error Messages

Error and trace messages appear in the Replication Server error log and, if the **logfile_path** trace option is set, the connector log.

## Incompatible Connector Version

The connector and Replication Server have been built using different canonical interface (CI) versions.

### *Symptom*

The trace message that appears in the Replication Server error log is similar to:

```
T. 2010/04/29 16:32:54. (17): Version [3.1] of Connector
   [ora.oci], is incompatible with CI version [3.2].
```

### *Explanation*

This trace occurs when a connector is built and based upon a version of the Canonical Interface (CI) specification that differs from that with which Replication Server is built. In the message above, the Oracle connector is compatible only with CI version 3.1 while Replication Server requires CI version 3.2.

### *Solution*

Ensure that the connector and CI have the same version. You may need to upgrade the connector to match the CI version used by Replication Server. You can find the CI (also seen as RCI) information by checking the version string within `libsybeconn.dll` (Windows) or `libsybeconn.so` (UNIX), for example:

```
strings libsybeconn.dll | grep Express
```

```
Sybase ExpressConnect-Library/15.5/P/RCI 3.2/
NT (IX86)/Windows 2003/1/DEBUG/Thu Apr 29 08:40:08 2010
```

## No Permission to Produce Connector for Unlicensed Feature

A valid license for a connector feature is unavailable.

### *Symptom*

The message reported in the Replication Server error log is similar to:

```
T. 2010/04/29 16:32:54. (17):  No permission to produce
   connector for unlicensed feature [REP_EC_ORA].
```

### *Explanation*

This error occurs when there is no valid SySAM license for the connector feature.

*Solution*
Buy the connector feature needed, such as ExpressConnect for Oracle. Install the feature product, and ensure that the license is correctly updated.

## Cannot Produce Connector from Factory

The factory library being loaded is missing or its name has changed.

*Symptom*
The message reported in the Replication Server error log is similar to:

```
T. 2010/04/29 16:32:54. (17):  Failed to produce
  Connector from factory using library [libsybora.dll].
```

*Explanation*
The message above indicates that the `libsybora.dll` factory was not able to produce the Oracle connector `libsyboraoci.dll`.

*Solution*
Ensure that the connector library is available in the directories defined in PATH (Windows) or LD_LIBRARY_PATH (UNIX).

## Loading of Connector Factory Failed

The factory library being loaded is missing or its name has changed.

*Symptom*
The message written in the Replication Server error log is similar to:

```
T. 2010/02/04 10:32:08. (22): Loading of Connector
   factory library [libsybora.dll] failed. Error=[126].
```

*Explanation*
On UNIX, the message describes the error in detail.

On Microsoft Windows, the error information is described in the Microsoft Developer Network (MSDN) Web site. To search for the error description in the MSDN Web site:

1. Go to *http://msdn.microsoft.com* and search for “system error codes.”
2. Look for the error code description in the “System Error Codes” page.

*Solution*
Ensure that the connector library is available in the directories defined in PATH (Windows) or LD_LIBRARY_PATH (UNIX).

# RepAgent Error Messages

RepAgent error messages are recorded in the Adaptive Server error log.

## Error 9202

Nested stored procedures are not allowed.

### *Symptom*

These error messages are reported in the Adaptive Server error log:

```
00:00000:00011:2006/06/13 16:26:05.12 server
Error: 9202, Severity: 20, State: 0
00:00000:00011:2006/06/13 16:26:05.12 server
RepAgent(5): Nested replicated stored procedure
detected. Transaction log may be corrupt. Please contact
SYBASE Technical Support. (current marker = 1372, 12)
00:00000:00011:2006/06/13 16:26:05.12 server Rep Agent
Thread for database 'pdb' (dbid = 5) terminated
abnormally with error. (major 92, minor 2)
```

### *Explanation*

A nested stored procedure is called from within another stored procedure. The stored procedure that calls the nested stored procedure is called the outer stored procedure.

When stored procedures with nested stored procedures are marked for replication with **sp_setrepproc**:

- The RepAgent shuts down.
- The RepAgent forwards only the outer stored procedure call to the Replication Server.
- An error message is reported in the Adaptive Server error log.

### *Solution*

Do not use nested replicated stored procedures. Instead:

1. Skip the nested stored procedure transaction.
   a. Find the page of the secondary truncation point:

      ```
      dbcc gettrunc
      ```

   b. Find a valid page after the nested stored procedure transaction:

      ```
      dbcc traceon(3604)
      dbcc pglinkage(dbid, pageid, 0,2,0,1)
      ```

      where *pageid* is the ID for the page you received by executing **dbcc gettrunc** in step a.

   c. Set a new secondary truncation point on the valid page after the nested stored procedure transaction:

      ```
      dbcc settrunc ('ltm', 'pageid', pageid)
      ```

where *pageid* is the ID for the page after the current page you retrieved using **dbcc pagelinkage** in step b.

**d.** Reset the locater:

```
rs_zeroltm
```

2. Reapply only the nested stored procedure transaction.
3. Restart RepAgent.

**Note:** This procedure may also cause data loss because manually setting the secondary truncation point to a later page in the log skips any begin transaction statements on skipped pages. Those transactions do not replicate.

## Error 9210

A network problem has been encountered.

### *Symptom*

These messages are reported in the Adaptive Server error log:

```
2006/09/07 09:41:26.34 RepAgent (10) Error: 9210,
Severity: EX_CMDFATAL, State: 1
Fatal error from CT-Lib.
```

```
Open Client Client-Library error: Error: 84083974,
Severity: 5 -- ct_results(): unable to get layer message
string: unable to get origin message string: Net-Library
operation terminated due to disconnect.
```

### *Explanation*

You see these messages when there are network problems; for example, when a RepAgent cannot initialize a Client-Library connection, or a Replication Server returned an unexpected error. Network problems can also cause RepAgent error 9212 (which is caused by a CT-Lib error).

### *Solution*

Fix the network problems and restart RepAgent. After you fix the network problems, execute **admin who_is_down** on the Replication Server. You see that the REP AGENT USER threads are down.

In Replication Monitoring Services (RMS), “suspect status” (yellow) appears for the Replication Server, while “normal” status (green) is shown for the RepAgent.

## Error 9215 (ASE 624)

The database log has been truncated.

### *Symptom*

This message appears in the Adaptive Server error log:

```
2006/10/17 11:57:19.34 RepAgent (10) Error: 9215,
Severity: EX_USER, State: 1
```

```
The Secondary Truncation Point is not valid. Please make
sure that the Secondary Truncation Point is valid, and
then restart the Rep Agent Thread for this database.
```

### *Explanation*

This error occurs when the log is truncated past the secondary truncation point and the RepAgent tries to reconnect to the database. RepAgent requests the log page number from the Replication Server and then accesses that page in the database. Because the log was truncated, the RepAgent will not start.

Attempts to retrieve rows from the page using the row identifier (RID) fail because the requested RID is a higher value than the last RID on the page. For example, truncating the log frees only those log pages that the RepAgent has sent to the Replication Server with confirmation.

### *Solution*

Because you truncated the log, you must reset the information about the database log in the `rs_locater` table within the Replication Server System Database (RSSD).

1. In the RSSD, execute:

   ```
   > rs_zeroltm primss, primdb
   > go
   ```

2. In the primary database, reactivate the secondary truncation point:

   ```
   > dbcc settrunc(ltm, valid)
   > go
   ```

3. Restart RepAgent.

Check for open transactions by comparing the output of **dbcc gettrunc** with the content of the `rs_locater` tables attribute locater. Truncating the transaction log may have caused orphaned transactions, which occur when the transaction's **begin** statement is transferred, but the **commit** or **rollback** statement is accidentally deleted during log truncation.

# Route Problems

Route problems occur when creating, altering, or dropping routes.

To troubleshoot a problem, obtain and analyze information from the Replication Server error logs, execute the **rs_helproute** stored procedure, or both.

### See also

- *Troubleshooting Overview* on page 5

## Routes

A route is one-way message stream from a source Replication Server to a destination Replication Server. Routes carry data modification commands, including those for Replication Server System Databases (RSSDs), and replicated functions or stored procedures between Replication Servers.

There are two types of routes—direct and indirect. A direct route sends messages directly from source to destination Replication Servers, with no intermediate Replication Servers. An indirect route sends messages through one or more intermediate Replication Servers.

The **rs_marker** stored procedure contains the full **create route**, **alter route**, or **drop route** command. Routes are created, altered, and dropped according to the following general procedure:

- The source Replication Server executes **rs_marker** against its RSSD.
- The RepAgent of the RSSD picks up the stored procedure execution and forwards it to the source Replication Server.
- The source Replication Server appends the stored procedure execution to its inbound queue and forwards it to the destination Replication Server. The identity of the destination Replication Server is obtained from parameters in the stored procedure.
- The destination Replication Server processes the stored procedure execution, which materializes subscriptions to the primary Replication Server RSSD `rs_routes`, `rs_subscriptions`, `rs_rules`, and `rs_locater` system tables.

### create route Process

The **create route** command designates the route to use for a connection from the current Replication Server to a remote Replication Server.

When a route is created:

- The source Replication Server connects to the destination Replication Server. The source Replication Server needs the correct user name and password to connect to the destination Replication Server.
- The source Replication Server sends a create route message to the destination Replication Server.
- The destination Replication Server receives this message and starts creating subscriptions to the source Replication Server system tables.
- When the system table subscriptions are created, the process is complete.

## drop route Process

The **drop route** command closes the route to another Replication Server.

By default, before dropping a route, the source Replication Server waits until the destination Replication Server has cleaned up its part of the route. Use the **with nowait** option to allow the source Replication Server to drop a route without informing the destination Replication Server. However, you must manually clean the destination Replication Server when it becomes available again.

**Warning!** Use the **with nowait** option only when the destination is temporarily irrecoverable and you must drop the route without waiting for the destination Replication Server to become available.

When dropping a route using the default settings:

- The source Replication Server drops all the system table subscriptions created by the destination.
- The source Replication Server sends a drop route protocol message to the destination Replication Server.
- After this message is delivered to the destination Replication Server (or to the next site, if the route is an indirect route), the source Replication Server drops the route by deleting the entry from its `rs_routes` system table and, for direct routes, deleting the outbound queue.
- The destination Replication Server receives the drop route message and cleans up its part of the route by deleting the locater for the source Replication Server from its `rs_locater` system table.

When dropping a route with the **with nowait** option:

- The source Replication Server drops all the system table subscriptions created by the destination.
- The source Replication Server cleans up the route by deleting the route entry from its `rs_routes` system table and, for direct routes, dropping the outbound queue.

### Cleaning Up the Destination Replication Server

Manually clean up the destination server's side of the route after running **drop route** with the **with nowait** option.

1. Verify that all the messages from the source Replication Server are gone from the Data Server Interface (DSI) queues.
2. From the `rs_subscriptions` system table, delete all the system table subscriptions entered by this site when the route was created.
3. Delete rows from the `rs_rules` system table for these subscriptions.
4. Clean up user subscription entries from the `rs_subscriptions` and `rs_rules` system tables.
5. Against each system table, execute:

   ```
   delete from system_table
       where prsid = sourceRS_siteid
   ```

   where:

   - *system_table* is the name of the system table (`rs_subscriptions` or `rs_rules`).
   - *sourceRS_siteid* is the site ID of the source Replication Server.

   For more information about the **delete** command, see the *Adaptive Server Enterprise Reference Manual: Commands*.
6. Delete the entry for this route from the `rs_routes` system table.
7. Delete the entry for the source Replication Server from the `rs_locater` system table.

## rs_helproute

The **rs_helproute** stored procedure displays the current state of a route and the subscriptions it is currently processing. Routes go through various states while being created, altered, or dropped.

Execute **rs_helproute** on the RSSD at the source or the destination Replication Server:

```
rs_helproute [replication_server]
```

where *replication_server* is an optional parameter indicating the name of a Replication Server. If you enter this parameter, information is given for routes to and from the specified Replication Server. Otherwise, information is provided for all routes to and from the current Replication Server.

For each route, **rs_helproute** returns:

- Route status

Status identifies the state of the route protocol. The status of a route is different at the source and destination Replication Servers. Analyze the route status at the source and destination Replication Servers to determine the problem.

- A list of system table subscriptions that are currently being processed

  Incomplete materialization or dematerialization of system table subscriptions is one of the most common problems.

  Because creating, altering, and dropping routes includes processing system table subscriptions, the list that **rs_helproute** returns helps you determine which subscriptions prevent you from proceeding to the next step in the process.

  If you are creating a route, the list includes the system table subscriptions that are being created. If you are dropping a route, the list includes the system table subscriptions that are being dropped.

  If no system table subscriptions are listed for a route, there are no problems with the system table subscriptions.

See the *Replication Server Reference Manual*.

# Problems with Creating Routes

Describes problems that can occur when creating a route.

## Common Problems

Lists and provides solutions to common problems that can occur when the create route command fails.

**Table 5. Common Problems when Creating a Route**

| Problem | Suggested Action |
|---|---|
| The destination Replication Server has no entry in its interfaces file for the source Replication Server. | Include this name in the interfaces file. |
| Cannot create a route that has just been dropped. | Wait until the destination Replication Server has cleaned up its route to the source Replication Server. Then the destination can accept the new route. You may need to resume this route from the source Replication Server—the source may have suspended this route after being refused connection by the destination. |

| Problem | Suggested Action |
|---|---|
| Canot create an indirect route; the log at the intermediate site reports `route is needed to reach the destination`. | Drop the route from the source to the destination using **drop route** with the **with nowait** clause. Next, create a route from the intermediate site to the destination site. Then, use **create route** with **set next site** to create an indirect route from the source to the destination. |

## Messages in the Error Log at the Source Replication Server

Lists and provides solutions to the error messages in the source Replication Server error log.

**Table 6. Messages at the Source Replication Server**

| Message | Explanation | Suggested Actions |
|---|---|---|
| RSI *destination name*: Login incorrect<br>RSI *destination name*: Trying to connect | The source Replication Server tries to connect to the destination Replication Server with the user name and password supplied with the **create route** command, but the destination Replication Server does not recognize this user name and password. | 1. Perform one of these tasks:<br>• Use **create user** or **alter user** to create or alter this user name and password at the destination Replication Server.<br>• Use **alter route** with **set username** to alter the user name and password for this route.<br>• Use **drop route** with **with nowait** to drop the route. Then re-create the route.<br>2. Use **resume route** to resume this route at the source Replication Server. |

| Message | Explanation | Suggested Actions |
|---|---|---|
| CM: Could not find interfaces entry for *destination name* | The Communications Manager (CM) in the source Replication Server reports that the destination Replication Server name is not found in the source Replication Server interfaces file. | Perform one of these tasks:<br>• Add the Replication Server name in the interfaces file, or<br>• If you did not intend to create a route to this destination, use **drop route** with **with nowait** to drop the route. |

## Output from rs_helproute at the Source Replication Server

Describes the **rs_helproute** output when it is executed at the source Replication Server while a route is being created.

**Table 7. rs_helproute Output at the Source Server**

| Message | Explanation | Suggested Actions |
|---|---|---|
| Route is being created. Source RS has not yet attempted to send the protocol message to the destination RS. | The source Replication Server cannot execute the **rs_marker** stored procedure against its Replication Server System Database (RSSD). | Restart the Replication Server. If restarting does not fix the route, check the RSSD and any related messages in the error logs. There may be a message in the error logs indicating some problem with the RSSD. |
| Route is being created. Either (1) RS is waiting for a route protocol message from the RSSD Replication Agent or (2) the RSSD Replication Agent inbound queue is not being processed. | A message may be stuck between the RSSD log and the Replication Server inbound queue. The sender Replication Server executed the stored procedure, but the RSSD Replication Agent cannot forward it to the Replication Server. | Check if the RSSD Replication Agent is experiencing problems sending messages to the Replication Server. If not, restart the RSSD Replication Agent. |
| | A message may be stuck between the inbound and outbound queues in the Replication Server. | Check if the Replication Server inbound queues are full or the distributor for the RSSD is not running. If neither of these problems exist, restart the RSSD Replication Agent. |

| Message | Explanation | Suggested Actions |
|---|---|---|
| Route is being created. Create route protocol message is waiting to be delivered to the destination. | The sending Replication Server is not processing its outbound queue. | Resume the route, if:<br>• The Replication Server Interface (RSI) experienced problems connecting to the destination Replication Server.<br>• The route was suspended.<br>Verify whether:<br>• The destination Replication Server is up.<br>• The RSI for the destination Replication Server running at the source Replication Server. |
| Route is being created. Destination has not yet finished creating system table subscriptions. | The destination Replication Server is still creating subscriptions to system tables. | Make sure the destination Replication Server is running. Resume the route, if:<br>• The RSI experienced problems connecting to the destination Replication Server.<br>• The source could not send messages to the destination.<br>• The system suspends the route due to an error or the user explicitly suspends it.<br>Check the list of system table subscriptions that still need to be created. |

### Output from rs_helproute at the Destination Server

Describes the **rs_helproute** output when executed at the destination Replication Server while a route is being created.

**Table 8. rs_helproute Output at the Destination Server**

| Message | Explanation | Suggested Actions |
|---|---|---|
| This site received the create route protocol message from the source RS and is in the process of creating system table subscriptions. | The destination Replication Server is completing its system table subscriptions. | Check the list of system table subscriptions that still need to be created. If the destination Replication Server is suspended in this state, there may be a problem with the system table subscriptions. |

## Troubleshooting Problems with Altering Routes

Fix problems encountered when using the **alter route** command.

1. Quiesce the replicated data system before executing **alter route**.

   See *Replication Server Administration Guide Volume 1*.
2. If you are changing a direct route to an indirect route, or if you are changing the intermediate site in an indirect route, create a route from the new intermediate site to the destination site.
3. Execute **rs_helproute** at the destination Replication Server to determine the success of an **alter route** command.

   Make sure that the output reflects the route change. If the route change did not succeed, follow the troubleshooting procedures in *Replication Server Troubleshooting Guide > Route Problems > Problems with Creating Routes > Output from rs_helproute at the Source Replication Server*.

#### See also

- *Output from rs_helproute at the Source Replication Server* on page 70

## Problems with Dropping Routes

Describes problems that can occur when dropping a route.

## Output from rs_helproute at the Source Replication Server

Describes the **rs_helproute** output when it is executed at the source Replication Server while route is being dropped.

The **rs_helproute** output for the **drop route** and **drop route with nowait** commands is the same, excep that, for **drop route with nowait** the status starts with:

```
Route is being dropped with nowait. ...
```

instead of:

```
Route is being dropped. ...
```

**Table 9. rs_helproute Output at the Source Replication Server**

| Message | Explanation | Suggested Actions |
|---|---|---|
| Route is being dropped. System table subscriptions are being dropped. | The destination Replication Server is trying to drop the system table subscriptions. | Check the list of system table subscriptions that still need to be dropped. |
| Route is being dropped. Waiting for the system table subscriptions to be dropped. | The system is in this state until the subscriptions are dropped. | The Replication Server should not be in this state for a long period of time. If it is, it may indicate a subscription problem. |
| Route is being dropped. Source RS has not yet attempted to send the drop route protocol message to the destination RS. | The sender Replication Server could not execute the **rs_marker** stored procedure against its Replication Server System Database (RSSD). | Restart the Replication Server.Check the RSSD and any related messages in the error logs. There may be a message in the error logs indicating some problem with the RSSD. |
| Route is being dropped. Drop route protocol message should be either in RSSD Replication Agent queue or in the inbound queue. | A message may be stuck between the RSSD log and the Replication Server inbound queue. The sender Replication Server executed the stored procedure, but the RSSD Replication Agent cannot forward it to the Replication Server. | Check if the RSSD Replication Agent is experiencing problems sending messages to the Replication Server. If not, restart the RSSD Replication Agent. |
| | A message may be stuck between the inbound and outbound queues in the Replication Server. | Check if the Replication Server inbound queues are full or the distributor for the RSSD is not running. If neither of these problems exist, restart the RSSD Replication Agent. |

| Message | Explanation | Suggested Actions |
|---|---|---|
| Route is being dropped. Drop route protocol message is waiting to be delivered to the destination. | The sender Replication Server is not processing its outbound queue. | Check that the destination Replication Server is operating. If the Replication Server Interface (RSI) cannot connect to the destination Replication Server, it may be necessary to resume the route. |

### See also

- *Replication Server Interface Problems* on page 103

## Output from rs_helproute at the Destination Server

Describes the **rs_helproute** output when it is executed at the destination Replication Server while route is being dropped.

**Table 10. rs_helproute Output at the Destination Server**

| Message | Explanation | Suggested Actions |
|---|---|---|
| Route is being dropped. | The route is being dropped by the source Replication Server.The destination Replication Server is not very active while a route is being dropped. All of the state changes are seen at the source Replication Server only. | Look at the output from **rs_helproute** at the source Replication Server. |

# Subscription Problems

Subscription problems occur when subscription materialization or dematerialization fails.

The replication process begins with subscription materialization, which is the process by which data is initially copied to the destination database. When you no longer want a subscription replicated to a destination database, dematerialize the subscription at the destination database. Dematerialization is the process by which data is deleted from the destination database.

**Note:** If you are using a Replication Agent, your subscription materialization process may differ from the process described here. See your Replication Agent documentation for the Replication Agent-specific subscription materialization process.

Subscription problem symptoms are easily identified and include:

- Materialization failure – no data in the subscription's replicate table at the destination database, invalid status for subscriptions at the primary and replicate Replication Servers, or materialization has been taking longer than is reasonable.
- Dematerialization failure – data still exists in the subscription's replicate table at the destination database, the status for subscriptions at the primary and replicate Replication Servers is invalid, or dematerialization has been taking longer than is reasonable.

Usually, the person who is conducting the materialization or dematerialization monitors the operation and reports any problems.

Some subscription problem symptoms are reported as error messages in the Replication Server error log. You might also need to use the diagnostic tools to identify subscription problem symptoms.

If a subscription problem caused the Data Server Interface (DSI) thread for the replicate database to abnormally terminate, restart it using the **resume connection** command.

### See also

- *Errors When DSI is Down or Suspended* on page 120
- *Troubleshooting Materialization Failures* on page 19
- *Troubleshooting Dematerialization Failures* on page 20

## Materialization Process

Materialization creates and activates subscriptions, and copies data from a primary database to a replicate database.

When publication subscriptions are materialized atomically, nonatomically, or incrementally, only one article subscription is processed at a time. When publication subscriptions are

materialized using the bulk or no materialization methods, all article subscriptions are processed together.

If, when you are materializing a new subscription, there are other materializing or dematerializing subscriptions for the same replication definition and replicate database, the new subscription is assigned a Pending status (the `recovering` column in `rs_subscriptions` is set to 2). If you execute **check subscription**, the subscription's status is returned as Pending at the replicate Replication Server and Invalid at the primary Replication Server.

If the primary Replication Server System Database (RSSD) is unavailable or any other problems occur, the `recovering` column in `rs_subscriptions` is set to 1 to denote that the subscription requires recovery.

Subscriptions with a Pending status are processed one at a time, in the order entered. The `request_date` column in the `rs_subscriptions` table indicates the time the subscription request was entered.

## Atomic Materialization

Atomic materialization is the default materialization method, and is invoked using the default version of the **create subscription** command.

**Note:** Atomic materialization is supported only for primary Adaptive Server.

If there are no other subscriptions for the same replication definition and replicate database, the subscription is defined at the primary Replication Server. If the primary Replication Server is unavailable or any other problems occur, the `recovering` column in the `rs_subscriptions` system table is set to 1 to denote that the subscription requires recovery.

After the definition stage is complete, the replicate Replication Server builds the materialization queue for the subscription. Use **admin who, sqm** to monitor this activity.

Replication Server executes the **rs_select_with_lock** function to select subscription rows from the primary site. After the materialization queue is built, the replicate Replication Server sends an activation request to the primary Replication Server. This request is passed through the primary database via the **rs_marker** system function. When the primary Replication Server receives the activation request, the subscription is marked valid. All updates following the request are sent to the replicate database if they match the subscription.

The primary Replication Server sends the activation request back to the replicate Replication Server, where it is inserted into the Data Server Interface (DSI) queue for the replicate database. When the DSI queue processes the request, the subscription status is changed to active.

The DSI thread also switches over to the materialization queue from its regular outbound queue for the site. The **admin who, dsi** command indicates which queue the DSI thread is processing. The materialization queue is applied to the replicate database. After the

materialization queue is processed, the subscription is marked valid and the materialization is complete.

If the replicate Replication Server runs out of stable queue segments:

1. Add a new partition to the replicate Replication Server.
2. If no partition is available, drop the subscription without purge.
3. Use nonatomic materialization.

**Note:** To estimate how much space a subscription needs, see the *Replication Server Design Guide*.

If the replicate database log cannot hold all of the data for the subscription in a single transaction:

1. Drop the subscription without purge.
2. Use the incremental version of atomic materialization.

## Nonatomic Materialization

Execute the **create subscription** command with the **without holdlock** option at the replicate Replication Server to create a subscription using the nonatomic materialization method. The subscription is saved in the replicate Replication Server System Database (RSSD). If there are no other subscription requests for the same replication definition and replicate database, the subscription is defined at the primary Replication Server.

**Note:** Nonatomic materialization is supported only for primary Adaptive Server.

After the definition stage is complete, the replicate Replication Server sends an activation request to the primary Replication Server. The replicate Replication Server immediately starts to build the materialization queue for the subscription. After the materialization queue is built, the subscription status becomes "Qcomplete". The replicate Replication Server sends a validation request to the primary Replication Server through the primary database. Use **admin who** to monitor this queue.

When the activation request arrives at the primary Replication Server, the subscription status becomes Active. All updates following the request are sent to the subscription.

The primary Replication Server returns the activation request to the replicate Replication Server. When the Data Server Interface (DSI) at the replicate Replication Server receives the request, the subscription status becomes Active and the transactions in the materialization queue are applied to the replicate database. If the materialization queue has not been built yet, the status returned by **check subscription** is Active, and not Qcomplete. If the materialization queue has been built, the status is Qcomplete and Active. The DSI thread switches over to the materialization queue from its regular outbound queue for the site. **admin who, dsi** shows which queue the DSI thread is processing.

After the contents of the materialization queue are applied to the replicate database, the subscription status becomes Materialized.

While the replicate Replication Server is applying inserts from the materialization queue, the validation request is moving from the primary database log, through the RepAgent, to the primary Replication Server.

Once the validation request arrives at the primary Replication Server, the subscription status becomes Valid at the primary Replication Server and the request is forwarded to the replicate Replication Server. The subscription status becomes Valid at the replicate Replication Server after the materialization queue is applied and the validation request reaches the beginning of the DSI queue.

**Warning!** The subscription data may be inconsistent at the replicate database from the time the DSI thread starts applying the materialization queue until the subscription is validated at the replicate Replication Server. This is the result of not using a holdlock while selecting the subscription data from the primary database. Once the subscription status becomes valid, however, the replicate data is consistent with the primary data.

## Direct Load Materialization

Use direct load materialization to materialize data between different kinds of primary and replicate databases.

Direct load materialization differs from other automatic materialization methods:

- No materialization queue is used with direct load materialization. Data is loaded directly from a primary table into a replicate table.
- Replication to other tables is not suspended during direct load materialization. DML operations on a primary table being materialized are stored in a catch-up queue and applied to the replicate table after the initial materialization phase. DML operations on a primary table that is not being materialized are replicated into the replicate table as the DSI receives them. Multiple tables can be concurrently materialized with direct load materialization.
- When subscription materialization stops due to an error, regular replication to other tables is not suspended.
- Multiple parallel threads can be used to load data from one primary table to its corresponding replicate table. You can tune this multi-threaded behavior with **max_mat_load_threads**.

The atomic and nonatomic materialization methods described here are only supported for an Adaptive Server primary. For an explanation of the different types and methods of materialization, see the *Replication Server Heterogeneous Replication Guide > Materialization > Types of Materialization* and the *Replication Server Administration Guide: Volume 1 > Manage Subscriptions > Subscription Materialization Methods*.

Direct load materialization can be used to materialize data:

- from Adaptive Server to HANA DB
- from Microsoft SQL Server to HANA DB
- from Oracle to HANA DB

- from DB2 UDB to HANA DB

**Note:** Direct load materialization is not supported for materializing data into an Adaptive Server database.

*Restrictions and Limitations for **create subscription***

- When the **direct_load** option is used, no other subscription can be created or defined at the same time for the same replicate table.
- The **direct_load** option is for subscriptions to table replication definitions only and is used with **without holdlock**. It cannot be used with **without materialization** or **incrementally**.
- The **user** and **password** options are used only with **direct_load**.
- You can only use the **direct_load** option against a physical database connection, not an alternate or logical connection. This is the case for both the primary connection—the connection specified in the replication definition—and the replicate connection—the connection specified in the subscription.
- The maintenance user of the primary database cannot be used in the **user** and **password** options to create subscriptions.
- You cannot use atomic materialization if the primary database is not Adaptive Server. For a primary database other than Adaptive Server, the only automatic materialization option supported is direct load.You cannot drop a subscription with the **with purge** option if the replicate database is not Adaptive Server.
- The **direct_load** option is available only if the replicate Replication Server site version and route version are 1571100 or later.
- You can use row filtering, name mapping, customized function strings and datatype mapping with subscriptions created using the **direct_load** option.
- Replication Server rejects any attempt to create a subscription with the **direct_load** option if the number of subscriptions being created has reached or exceeded **num_concurrent_subs**.

## Bulk Materialization

The bulk materialization method involves manually transferring subscription data between databases. Use this method when a subscription is too large to copy through the network, or when other subscription methods are not supported for the primary or replicate database.

Use **define subscription** to add the subscription to the Replication Server System Database (RSSD) for the primary and replicate Replication Servers.

After the subscription is defined, use **activate subscription** to set the subscription status to Activating at the replicate Replication Server and to send the activation request to the primary Replication Server.

When the primary Replication Server receives the activation request, it changes the subscription status to Active, returns the activation request to the replicate Replication Server, and begins sending updates for the subscription to the replicate Replication Server. When the

replicate Replication Server receives the activation request, it changes the subscription status to Active at the replicate Replication Server.

If you specified **with suspension** for the **activate subscription** command, the Data Server Interface (DSI) thread is suspended when the activation request is received at the replicate Replication Server. At this stage, you load the replicate database. See the *Replication Server Reference Manual*. After loading, execute the **resume connection** command to continue applying transactions from the DSI.

After the subscription is activated, use **validate subscription** to verify that the data at the replicate data is consistent with the primary data at the replicate Replication Server and to complete bulk materialization. The subscription status is changed to Validating. The replicate Replication Server forwards the validation request to the primary Replication Server.

When the primary Replication Server receives the validation request, it changes the status at the primary to Valid and returns the validation request to the replicate Replication Server. When the replicate Replication Server receives the validation request, the subscription status becomes Valid at both sites, and the bulk materialization is complete.

## Dematerialization Process

Dematerialization removes subscriptions and, optionally, data from the replicate database. It also removes subscription information from the Replication Server System Database (RSSD) at the primary and replicate sites.

When you execute **drop subscription** at the replicate Replication Server, the subscription status becomes Dematerializing in the replicate Replication Server `rs_subscriptions` table. If there are any other materializing or dematerializing subscriptions for the same replication definition and replicate database, the `recovering` column in the `rs_subscriptions` table is set to 2 to indicate that the dematerialization request is pending. The **check subscription** command indicates that the subscription has a Dematerializing or Pending status at the replicate Replication Server.

Subscription creation and drop requests are processed one at a time in the order entered. The `request_date` column in `rs_subscriptions` stores the time a request was entered.

There are two methods for dematerializing subscriptions:

- **with purge** – deletes the subscription data from the replicate database.
- **without purge** or bulk – does not delete the subscription data from the replicate database.

Both methods go through two stages:

- Dematerialization – stops sending updates for the subscription to the replicate database and, optionally, deletes the subscription data from the replicate database.
- Removal – deletes the subscription from the system tables of the primary and the replicate Replication Server.

## with purge Dematerialization

The dematerialize with purge method is invoked by using the **with purge** option of the **drop subscription** command. The subscription status is set to Dematerializing at the replicate Replication Server, and a drop request is forwarded to the primary Replication Server.

**Note:** The with **purge option** can only be used with the **drop subscription** command if the replicate database is Adaptive Server.

When the primary Replication Server receives the drop request, it stops sending updates for the subscription to the replicate Replication Server. The subscription status is changed to Dematerializing at the primary Replication Server and a drop request is returned to the replicate Replication Server.

When the replicate Replication Server receives the drop request, it creates a dematerialization queue and starts a dematerialization thread to fill the queue. The dematerialization thread selects subscription data from the replicate database and puts a row delete operation into the dematerialization queue for each row that does not belong to any other active subscription.

While the dematerialization thread is writing to the dematerialization queue, the Data Server Interface (DSI) thread switches to the dematerialization queue and begins applying the deletes to the replicate database.

If the dematerialization thread encounters an error, the entire dematerialization process restarts and the same error may recur. To avoid this problem, suspend the connection to the replicate database. Then resume it using the **skip transaction** option. This resumes the connection with the second transaction in the queue. The first transaction is written to the exceptions log.

When dematerialization has completed, the subscription status at the replicate Replication Server is changed to Removing. The replicate Replication Server logs in to the primary Replication Server and requests the deletion of the subscription from its system tables. When that request succeeds, the replicate Replication Server removes the subscription from its own system tables and the dematerialization process is complete.

If the replicate Replication Server runs out of stable queue segments while dematerializing using the **with purge** option:

1. Add a new partition to the replicate Replication Server. If no partition is available, consider using bulk dematerialization.
2. If you have difficulty deleting subscription data, drop the subscription using the **without purge** option.
3. If you have difficulties because of permissions, a user with the appropriate permission should issue the **drop subscription** command.

### Bulk Dematerialization

Bulk dematerialization is invoked using the **without purge** option of the **drop subscription** command. The subscription status becomes Dematerializing at the replicate Replication Server and a drop request is forwarded to the primary Replication Server.

When the primary Replication Server receives the drop request, it stops sending updates for the subscription to the replicate Replication Server. The subscription status becomes Dematerializing at the primary Replication Server and a drop request is returned to the replicate Replication Server.

When the replicate Replication Server receives the drop request, the subscription status is changed to Removing at the replicate. The replicate Replication Server logs in to the primary Replication Server and requests that it delete the subscription from its system tables. When that request has succeeded, the replicate Replication Server removes the subscription from its own system tables and the dematerialization is complete.

## check subscription

The **check subscription** command returns messages that describe the status of the materialization process. Use this command if all of the servers and Replication Server threads are running but the subscription is not being created or dropped.

Execute **check subscription** at the primary and replicate Replication Servers. If the primary and replicate Replication Servers are the same Replication Server, execute **check subscription** only once; **check subscription** shows separate primary and replicate status messages.

Use **check subscription for publication** to find out the materialization status of publication subscriptions. For atomic, nonatomic, or incremental publication subscriptions, **check subscription for publication** shows the status of the current article subscription being processed; whereas for bulk or no materialization, the status of all article subscriptions is shown because all article subscriptions are processed together.

### Materialization Status

The materialization status and messages returned when executing **check subscription** at the primary and replicate Replication Servers.

**Table 11. Materialization Status at Replicate Replication Server**

| Status | Message |
|---|---|
| Invalid | *subscription name* doesn't exist. |

| Status | Message |
|---|---|
| Removing | `REMOVING subscription subscription name from system tables at the Replicate.` |
| Dematerializing | `Subscription subscription name is DEMATERIALIZING at the Replicate.` |
| Valid | `Subscription subscription name is VALID at the Replicate.` |
| Validating | `Subscription subscription name is VALIDATING at the Replicate.` |
| Materialized | `Subscription subscription name has been MATERIALIZED at the Replicate.` |
| Materialized (subscriptions created with the **direct_load** option) | `Subscription subscription name progress: catchup, xx% done, xxxxx commands remaining.` |
| Active | `Subscription subscription name is ACTIVE at the Replicate.` |
| Active (subscriptions created with the **direct_load** option) | `Subscription subscription name progress: initial loading, xx% done, xxxxx commands remaining.` |
| Activating | `Subscription subscription name is ACTIVATING at the Replicate.` |
| Qcomplete and Active | `Subscription subscription name is ACTIVE at the Replicate and Materialization Queue has been completed.` |
| Qcomplete | `Materialization Queue for Subscription subscription name has been completed.` |
| Active and Qcomplete | `Subscription subscription name is ACTIVE at the Replicate, but Materialization Queue for it has not been completed.` |
| Defined | `Subscription subscription name has been defined at the Replicate.` |
| Error | `Subscription subscription name has experienced an unrecoverable error during Materialization or Dematerialization. Please consult the error log for more details.` |

| Status | Message |
|---|---|
| Pending | `Other subscriptions are being created or dropped for the same replication definition/database. Subscription` *`subscription name`* `will be processed when previous requests are completed.` |
| Recovering | `Subscription` *`subscription name`* `has experienced a recoverable error during Materialization or Dematerialization. It will be recovered by Subscription Daemon (dSub).` |
| Dropping | `Subscription` *`subscription name`* `is being dropped.` |

**Table 12. Materialization Status at Primary Replication Server**

| Status | Message |
|---|---|
| Invalid | *`subscription name`* `doesn't exist.` |
| Dematerializing | `Subscription` *`subscription name`* `is DEMATERIALIZING at the Primary.` |
| Valid | `Subscription` *`subscription name`* `is VALID at the PRIMARY.` |
| Active | `Subscription` *`subscription name`* `is ACTIVE at the PRIMARY.` |
| Activating | `Subscription` *`subscription name`* `is ACTIVATING at the PRIMARY.` |
| Defined | `Subscription` *`subscription name`* `has been defined at the PRIMARY.` |

## Materialization Problems

Provides information for troubleshooting atomic, nonatomic, bulk, and other common materialization problems.

## Incorrect or Missing Login Account and Permissions

Many problems with materialization are due to incorrect or missing permissions in primary or replicate databases. Error messages that identify these problems are reported in the replicate Replication Server error log.

*User Requirements for Creating Subscriptions*

Verify that the login accounts for the user creating a subscription meet these requirements:

- The user's login name and password are the same at the replicate Replication Server, the primary Replication Server, and the primary data server.
- The user has been added to the primary database.
- The user has **select** permission on the primary table.
- The user has **execute** permission on the **rs_marker** stored procedure.

  This requirement applies to a configuration that uses RepAgent. If you are using a Replication Agent for a non-Adaptive Server data server, you may want to check your Replication Agent documentation for a similar requirement

  The default installation scripts grant execute permission on **rs_marker** to "public," so this should not be an issue unless you did not use the installation scripts.
- The user has at least **create object** permission at the replicate Replication Server.
- The user has at least **primary subscribe** permission at the primary Replication Server.
- In the case of direct load materialization from a non-Adaptive Server primary, the user ID and password supplied in the Replication Server **create subscription** command are for the Replication Agent admin user, not the primary non-Adaptive Server database user.

If you cannot give the user the required accounts and permissions, drop the subscription without purge and have a different user create it.

*Maintenance User Permissions*

Make sure that the maintenance user for the replicate database has **update**, **delete**, **insert**, and **select** permission on the replicate table. Lack of **update** permission causes errors in the replicate database when the Data Server Interface (DSI) attempts to apply transactions in the materialization queue

Depending on error action assignments, some transactions may be rejected and recorded in the exceptions log.

## Schema Inconsistency

Materialization problems can result from schema inconsistencies at the primary database, Replication Server, and the replicate database. Error messages that identify these problems are reported in the replicate Replication Server error log.

*Conflicting Table or View in the Primary or Replicate Database*
A table or view with the same name and columns as the replication definition has been created at the primary or replicate database. If you have created custom function strings, make sure they match the replicate table at the replicate database.

This can cause errors at the primary database as a result of executing **select** during subscription materialization.

At the replicate database, this may cause errors when the Data Server Interface attempts to apply transactions in the materialization queue. Some transactions, depending on the error action assignments, may be rejected and placed into the exceptions log.

*Missing rs_select Function String*
If the primary database does not use one of the system-provided function-string classes (`rs_sqlserver_function_class`, `rs_default_function_class`, `rs_db2_function_class`) or a function-string class that is a child of `rs_default_function_class` or `rs_db2_function_class`, make sure there are function strings for the **rs_select** function that match the **where** clause of the **create subscription** or **define subscription** command.

*The Replication Definition or Replicate Table Column length is Too Short*
Make sure column lengths are long enough in the replication definition and replicate table.

## Missing interfaces File Entries

Login attempts may fail due to missing `interfaces` file entries. The message that identifies this materialization problem appears in the replicate Replication Server error log.

The `interfaces` file used by the replicate Replication Server should contain an entry for the primary Replication Server and an entry for the primary data server.

## Atomic Materialization Problems

Fix atomic materialization problems based on the status returned by **check subscription**.

**Table 13. Atomic Materialization Problems**

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Defined/ Pending | Invalid | Waiting for other subscriptions for the same replication definition and replicate database to be processed. | Check for other subscriptions being created and dropped for the same replication definition and replicate database.<br>If there are no other subscriptions, wait five minutes. |
| Defined/ Recovering | Invalid | Cannot connect to the primary Replication Server to define the subscription. | Check the replicate Replication Server error log for messages.<br>Make sure the user creating the subscription has the same login name and password at the primary Replication Server and the replicate Replication Server. Make sure the user has at least **primary subscribe** permission at the primary Replication Server. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Defined/ Recovering | Defined | Cannot build the materialization queue. | Verify that the primary data server is running.<br>Make sure the user creating the subscription has the same login name and password at the primary data server and replicate Replication Server, **select** permission on the primary table, and **execute** permission for **rs_marker**.<br>Use **admin disk_space** to investigate whether the replicate Replication Server is out of stable queue segments.<br>Use **admin who, dsi** and **admin who, sqm** to monitor the queues. |
| Defined | Defined | Building materialization queue. | Building the queue may take some time. Wait until this process is complete.<br>Use **admin who, sqm** to monitor materialization.<br>Use **admin disk_space** to investigate whether the primary Replication Server is out of stable queue segments.<br>Check if rows are being selected using a holdlock at the primary database. |
| Qcomplete | Defined | Waiting for the activation request to get to the primary Replication Server.<br>Materialization queue has been built. | Verify that the RepAgent for the primary database is running.<br>Investigate whether the primary Replication Server is out of stable queue segments. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Qcomplete/ Recovering | Defined/ Valid | The replicate Replication Server stopped after the materialization queue was completed. | Wait. The status should return to Qcomplete at the replicate Replication Server. |
| Qcomplete | Activating | The activation request is being processed at the primary Replication Server. | Wait. The status at the primary Replication Server should change to valid. |
| Qcomplete | Valid | The subscription is complete at the primary Replication Server.<br>Waiting for activation request or applying the materialization queue at the replicate Replication Server. | Check the route between the primary and the replicate Replication Server.<br>Check the replicate Replication Server Data Server Interface (DSI) thread for the replicate database.<br>Investigate whether the replicate Replication Server ran out of queue segments.<br>Use **admin who, dsi** and **admin who, sqm** to monitor the queues. If **admin who, dsi** shows that the DSI is processing the materialization queue, wait until the queue has been processed. The length of the wait depends upon the size of the queue. |
| Valid | Valid | Complete. | None. |

## Nonatomic Materialization Problems

Fix nonatomic materialization problems based on the status returned by **check subscription**.

**Table 14. Nonatomic Matarialization Problems**

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Defined/ Pending | Invalid | The replicate Replication Server is waiting for other subscriptions for the same replication definition and replicate database to be created or dropped. | Look for other subscriptions being created or dropped for the same replication definition and replicate database.<br>If there are no other subscriptions, wait for five minutes. |
| Defined/ Recovering | Invalid | The replicate Replication Server cannot connect to the primary Replication Server to define the subscription. | Check the replicate Replication Server error log for messages.<br>Make sure the user creating the subscription has the same login name and password at the primary Replication Server and the replicate Replication Server. The user should have at least **primary subscribe** permission on the primary Replication Server. |
| Defined | Defined | The replicate Replication Server is building the materialization queue.<br>The primary Replication Server is waiting for the activation request. | Verify that the materialization queue is being built.<br>Verify that the primary Replication Server is up and that the Stable Queue Manager (SQM), Stable Queue Transaction (SQT), and Distributor (DIST) threads for the primary database are running. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Defined/ Recovering | Defined | The replicate Replication Server cannot build the materialization queue or cannot send the activation request to the primary Replication Server. | Verify that the primary data server is running.<br>Make sure the user creating the subscription has the same login name and password at the primary data server, **select** permission on the primary table, and **execute** permission for **rs_marker**.<br>Investigate whether the replicate Replication Server is out of stable queue segments.<br>Verify that the primary Replication Server is running and that the SQM, SQT, and DIST threads for the primary database are running.<br>Investigate whether the primary Replication Server is out of segments. |
| Defined | Active | The replicate Replication Server is building the materialization queue.<br>The primary Replication Server has received the activation request and sent it to the replicate Replication Server. | Verify that the materialization queue is being built.<br>Verify that the connection between the primary and replicate Replication Server is up.<br>Verify that the Data Server Interface (DSI) thread for the replicate database is running. |
| Qcomplete | Defined | Waiting for the activation request to be processed by the distributor at the primary Replication Server. | Investigate whether the primary Replication Server is out of stable queue segments.<br>Verify that the primary Replication Server is up and the SQM, SQT, and DIST threads for the primary database are running. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Qcomplete/ Recovering | Defined | The replicate Replication Server was recycled.<br>Waiting for the activation request to be processed by the distributor at the primary Replication Server. | Wait. The state at the replicate Replication Server should soon change to Qcomplete. |
| Qcomplete | Active | The primary Replication Server received and processed the activation request. Now, this Replication Server is waiting for the validation request.<br>The replicate Replication Server is waiting for the activation request. | Check the route from the primary Replication Server to the replicate Replication Server.<br>Verify that the DSI thread for the replicate database is running.<br>Investigate whether the replicate Replication Server has run out of queue segments. |
| Qcomplete | Valid | The subscription is done at primary Replication Server.<br>The replicate Replication Server is waiting for the activation request. | Check the route between the primary Replication Server and the replicate Replication Server.<br>Check the DSI to the replicate database.<br>Investigate whether the replicate Replication Server has run out of queue segments. |
| Active and not Qcomplete | Active | The replicate Replication Server is applying and building the materialization queue at same time.<br>The primary Replication Server is waiting for the validation request. This request is not sent until the queue is complete at the replicate Replication Server. | Verify that the DSI thread is processing the materialization queue.<br>Investigate whether the replicate Replication Server is out of stable queue segments.<br>Use **admin who, sqm** to monitor the materialization queue activity. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Qcomplete and Active | Active | The primary Replication Server is waiting for the validation request.<br>The replicate Replication Server is applying the materialization queue. | Verify that the DSI is processing the materialization queue.<br>Use **admin who, sqm** to monitor the materialization queue activity.<br>Verify that the primary data server, the RepAgent for the primary database and the RepAgent User thread in the primary Replication Server are running.<br>Verify that the primary Replication Server is running and that the SQM, SQT, and DIST threads for the primary database are running.<br>Investigate whether the primary Replication Server is out of segments. |
| Qcomplete and Active | Valid | The primary Replication Server received the validation request.<br>The replicate Replication Server is applying the materialization queue. | Verify that the DSI thread is processing the materialization queue.<br>Use **admin who, sqm** to monitor the materialization queue activity.<br>Check the route between the primary Replication Server and the replicate Replication Server. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Materialized | Active | The primary Replication Server is waiting for the validation request.<br>The replicate Replication Server has finished applying the materialization queue. | Investigate whether the primary Replication Server is out of segments.<br>Verify that the primary data server, the RepAgent for the primary database, and the RepAgent User thread at the primary Replication Server are running.<br>Verify that the SQM, SQT, and DIST threads for the primary database are running. |
| Materialized | Valid | The primary Replication Server received the validation request.<br>The replicate Replication Server has finished applying the materialization queue. | Check the route between the primary Replication Server and the replicate Replication Server.<br>Verify that the DSI for the replicate database is running.<br>Investigate whether the replicate Replication Server has run out of stable queue segments. |
| Valid | Valid | Complete. | None. |

## Direct Load Materialization Problems

Fix direct load materialization problems that occur during the definition, activation, and validation phases based on the status returned by **check subscription**.

**Table 15. Direct Load Materialization Problems**

| Status at Replicate Replication Server | Problem | Suggested Actions |
|---|---|---|
| Materialized | A subscription stays at the MATERIALIZED state. | Verify that Replication Agent is running. Make sure that the replication path is correct, either by replicating to another subscription which is already in the VALID state or by using the **rs_ticket** stored procedure. |
| Error | The primary or replicate replication definition datatype or column length is inconsistent. | Check the replicate Replication Server log, and check the datatype and column lengths. Correct the problem, drop the subscription, delete data at the replicate, and recreate the subscription. |

## Bulk-Materialization Problems

Fix bulk-materialization problems that occur during the definition, activation, and validation phases based on the status returned by **check subscription**.

**Table 16. Bulk Materialization Problems—Definition Phase**

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Defined/ Pending | Invalid | Waiting for other subscriptions for the same replication definition and replicate database. | Look for other subscriptions for the same replication definition and database.<br>If there are no other subscriptions, wait for five minutes. |
| Defined/ Recovering | Invalid | Cannot connect to the primary Replication Server to define the subscription. | Check the replicate Replication Server error log for messages.<br>Make sure the user creating the subscription has the same login name and password at the primary Replication Server and the replicate Replication Server. |
| Defined | Defined | Definition cycle is complete. | Activate the subscription. |

**Table 17. Bulk Materialization Problems—Activation Phase**

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Activating/ Recovering | Defined | Cannot send the activation request to the primary Replication Server. | Verify that the primary Replication Server is running and the Stable Queue Manager (SQM), Stable Queue Transaction (SQT), and Distributor (DIST) threads for the primary database are running.<br>Check the error log in the replicate Replication Server.<br>Investigate whether the primary Replication Server has run out of segments. |
| Activating | Active | The primary Replication Server received the activation request and returned it to the replicate Replication Server.<br>The replicate Replication Server is waiting for the activation request. | Verify the connection between the primary Replication Server and the replicate Replication Server.<br>Verify that the Data Server Interface (DSI) thread for the replicate database is running. |
| Active | Active | The activation stage is complete. | Validate the subscription.<br>If you specified the **with suspension** option for the **activate subscription** command, you may now load the replicate database. |

**Table 18. Bulk Materialization Problems—Validation Phase**

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Validating/ Recovering | Active | Cannot send the validation request to the primary Replication Server. | Verify that the primary Replication Server is running. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Validating | Valid | The primary Replication Server received the validation request.<br>The replicate Replication Server is waiting for the validation request. | Verify the route between the primary and replicate Replication Server.<br>Verify the DSI is running.<br>Verify that the replicate Replication Server has sufficient queue segments. |
| Valid | Valid | Complete. | None. |

## Dematerialization Problems

Fix dematerialization problems based on the status returned by **check subscription**.

Check the error logs for all of the servers involved for specific information. Also check that the user who is dropping the subscription has the same login name and password at the replicate and primary Replication Servers, and, if you used the **with purge** option, that the maintenance user for the replicate database has **select**, **delete**, and **update** permissions on the replicate table.

**Table 19. Dematerialization Problems—with purge Option**

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Dematerializing/ Pending | N/A | Waiting for other subscriptions for the same replication definition and replicate database. | Look for other subscriptions being created or dropped for the same replication definition and database.<br>If no other subscription operations are in process, wait for five minutes. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Dematerializing/ Recovering | N/A | Cannot connect to the primary Replication Server to drop the subscription. | Check replicate Replication Server error log for messages.<br>If the user dropping the subscription does not have the same login and password at the primary and replicate Replication Servers, then:<br>• Provide the user with a Replication Server account that has the same login and password at the primary and replicate, or<br>• Have a different user with the appropriate permissions drop the subscription.<br>Also, the user must have at least **primary subscribe** privileges at the primary Replication Server. |
| Dematerializing | N/A | The primary Replication Server is waiting for the drop request. | Investigate whether the primary Replication Server is out of queue segments.<br>Verify that the primary Replication Server is running and that the Stable Queue Manager (SQM), Stable Queue Transaction (SQT), and Distributor (DIST) threads for the primary database are running. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Dematerializing | Dematerializing | The primary Replication Server processed the drop request and sent it to the replicate Replication Server.<br>The replicate Replication Server is waiting for the drop request. | Check the route between the primary Replication Server and the replicate Replication Server.<br>Check the Data Server Interface (DSI) thread for the replicate database.<br>Investigate whether the replicate Replication Server has run out of queue segments. |
| Dematerializing/ Recovering | Dematerializing | The primary Replication Server processed the drop request and sent it to the replicate Replication Server.<br>The replicate Replication Server failed to create and process the dematerialization queue. | Investigate whether the replicate Replication Server has run out of queue segments.<br>Verify that the maintenance user has **select** privileges on the replicate table.<br>Check the DSI thread for the replicate database. |
| Removing/ Recovering | Dematerializing | The subscription has dematerialized at the replicate Replication Server, but the replicate Replication Server could not log in to primary Replication Server to remove the subscription from the system tables. | Verify that the primary Replication Server is running.<br>If the user dropping the subscription does not have the same login and password at the primary and replicate Replication Servers, then:<br>• Provide the user with a Replication Server account that has the same login and password at the primary and replicate, or<br>• Have a different user with the appropriate permissions drop the subscription. |

| Status at Replicate Replication Server | Status at Primary Replication Server | Subscription State | Suggested Actions |
|---|---|---|---|
| Removing | Dematerializing | The primary Replication Server is deleting the subscription.<br>The replicate Replication Server is waiting for the primary Replication Server to finish. | Wait. |
| Removing | Invalid | The subscription has been removed from the primary Replication Server.<br>The replicate Replication Server will remove the subscription next. | Wait. |
| Invalid | Invalid | The subscription has been dropped. | None. |

**Table 20. Dematerialization Problems—without purge Option**

| Replicate Status | Primary Status | Subscription State | Suggested Actions |
|---|---|---|---|
| Dematerializing/ Pending | N/A | Waiting for other subscription requests for the same replication definition and replicate database. | Check for other subscription being created or dropped for the same replication definition and database.<br>If there are no other subscriptions, wait for five minutes. |

| Replicate Status | Primary Status | Subscription State | Suggested Actions |
|---|---|---|---|
| Dematerializing/ Recovering | N/A | Cannot connect to the primary Replication Server to drop the subscription. | Check the replicate Replication Server error log for messages.<br>Make sure the user who created the subscription has the same login name and password at the primary Replication Server and the replicate Replication Server. The user should also have at least **primary subscribe** privileges. |
| Dematerializing | N/A | The primary Replication Server is waiting for the drop request. | Determine whether the primary Replication Server has run out of queue segments.<br>Verify that the primary Replication Server is up and that the SQM, SQT, and DIST threads for the primary database are running. |
| Dematerializing | Dematerializing | The primary Replication Server processed the drop request and sent it to the replicate Replication Server.<br>The replicate Replication Server is waiting for the drop request. | Check the route between the primary Replication Server and the replicate Replication Server.<br>Check the DSI thread for the replicate database.<br>Determine whether the replicate Replication Server ran out of queue segments. |
| Dematerializing/ Recovering | Dematerializing | The primary Replication Server processed the drop request and returned it to the replicate Replication Server.<br>The replicate Replication Server terminated abnormally. | Wait for the subscription daemon to reset the recovering flag. |

| Replicate Status | Primary Status | Subscription State | Suggested Actions |
|---|---|---|---|
| Removing/ Recovering | Dematerializing | The replicate Replication Server could not log in to the primary Replication Server to delete the subscription from the system tables. | Verify that the primary Replication Server is running. |
| Removing | Dematerializing | The primary Replication Server is deleting the subscription from the system tables.<br>The replicate Replication Server is waiting for the primary Replication Server to finish. | Wait. |
| Removing | Invalid | The subscription is removed from the primary Replication Server.<br>The replicate Replication Server will remove the subscription next. | Wait. |
| Invalid | Invalid | The subscription has been dropped. | None. |

# Replication Server Interface Problems

Replication Server Interface (RSI) allows two Replication Servers to exchange transactions across a route. Usually, RSI problems occur when Replication Server attempts to connect to another Replication Server or when the locater is invalid.

The RSI consists of:

- A stable queue and thread at the primary Replication Server. The thread reads the stable queue and writes to the network.
- A thread at the replicate Replication Server. This thread reads from the network and writes into outbound stable queues.

Although many errors may occur when a primary Replication Server loses its connection to the replicate Replication Server, none of these errors should cause a loss of data or cause the RSI to be suspended. The primary thread continuously attempts to connect to the replicate Replication Server until the connection is reestablished.

Losing the connection terminates the RSI thread at the replicate Replication Server. The thread at a replicate Replication Server exists only while a connection exists. Executing **admin who** in the replicate Replication Server shows that the primary Replication Server is no longer logged in as an RSI source. When the primary Replication Server reestablishes the connection to the replicate Replication Server, data is replicated across the route to the replicate Replication Server.

### See also

- *Common Error Messages* on page 35
- *Troubleshooting Overview* on page 5

## Incorrect RSI User Login Name or Password

The primary Replication Server logs in to the replicate Replication Server using the Replication Server Interface (RSI) login name and password. The RSI user login name and password are created by **rs_init** and are also required when you create a route to the replicate Replication Server.

If there is a problem with the login name and password at the replicate Replication Server, the primary Replication Server logs one of these errors:

```
I. 2006/06/23 14:29:43. RSI: Trying to connect to
'eastRS'.
E. 2006/06/23 14:29:43. ERROR #1028 RSI(eastRS) -
seful/cm.c(3463)   Message from server: Message: 14021,
State 0, Severity 12 -- 'Invalid login attempted by user
'Rep_Server_rsi''.
```

```
E. 2006/06/23 14:29:44. ERROR #1027 RSI(eastRS)-
seful/cm.c(3463)   Open Client Client-Library error:
Error: 67175468, Severity 4 -- 'ct_connect(): protocol
specific layer: external error: The attempt to connect
to the server failed.'.
E. 2006/06/23 14:29:44. ERROR #13045 RSI(eastRS) -
seful/cm.c(3467)   Failed to connect to server 'eastRS'
as user 'REP_Server_rsi'. See CT-Lib and/or server error
messages for more information.
E. 2006/06/23 14:29:44. ERROR #4044 RSI(eastRS) -
i/rsiint.c(329)   RSI for 'eastRS': Shutting down due
to an exception.
```

To repair this problem:

1. Retrieve the RSI user login name and password from the Replication Server System Database (RSSD) of the primary Replication Server by using **isql** to execute this query on the RSSD:

   ```
   select username, password
    from rs_maintusers, rs_sites
    where name = remote_RS_name and destid = id
   ```

   where:

   - *remote_RS_name* is the destination Replication Server name.
   - *id* is the site ID of the destination Replication Server.

   If you have password encryption, you cannot access the password by executing a query.
2. Perform one of these actions:
   - Use **alter route** to change the password for the primary Replication Server to the RSI user password, which you retrieved in the previous step.
   - Use **alter user** to change the password for the login name at the replicate Replication Server. Then use **resume route** to resume the route.

## Incorrect User Permissions at Replicate Replication Server

The Replication Server Interface (RSI) login must have **connect source** or **sa** permission on the replicate Replication Server. If the login lacks the appropriate permission, an error message is sent to the primary Replication Server error log.

To repair this problem:

1. Grant **connect source** permission to the login at the replicate Replication Server.
2. Resume the route at the primary Replication Server.

## Invalid RSI Locater

An invalid Replication Server Interface (RSI) locater indicates that the RSI locaters at the primary and replicate Replication Servers are not identical, and that the primary Replication Server deleted messages from its stable queue at an incorrect location.

An RSI locater identifies the last message that the replicate Replication Server stable queue received from the primary Replication Server. The RSI uses the RSI locater in the `rs_locater` system table to delete messages in the stable queues. The replicate Replication Server updates the RSI locater in the `rs_locater` system table with the last messages that the replicate Replication Server added to its stable queues. Periodically, the primary Replication Server requests the RSI locater from `rs_locater` at the replicate Replication Server. The primary Replication Server deletes its stable queue messages up to and including the message identified by the RSI locater—these are the messages that the replicate Replication Server has already received.

**Note:** The RSI locater is different from the locater that is used with database log truncation.

If the RSI locaters at the primary and replicate Replication Servers are not identical, the primary Replication Server might delete messages from its stable queue at an incorrect location and invalidate the RSI locater at the replicate Replication Server.

If RSI locaters are not identical:

1. Make sure that the replicate Replication Server is not running.
2. Set the replicate Replication Server RSI locater to 0 by executing this command against the replicate Replication Server System Database (RSSD):

```
update rs_locater set locater=0x0
 where sender = primary_replication_server
```

### See also

- *Resetting the Database Log Locater* on page 130

# RepAgent Problems

RepAgent errors can be caused by problems with the Adaptive Server, Replication Server, or RepAgent. RepAgent records messages in the Adaptive Server error log. These messages identify the server that caused the error so you can diagnose and correct the problem.

RepAgent retrieves the data for primary objects from the Adaptive Server log and converts the log record information into Log Transfer Language (LTL) commands. These commands are sent to the primary Replication Server for distribution and replication. RepAgent also coordinates database log truncation with the Adaptive Server and primary Replication Server.

For more information about how the RepAgent processes errors, see the *Replication Server Administration Guide Volume 1*.

**See also**

- *Common Error Messages* on page 35
- *Troubleshooting Overview* on page 5

## Problems when Starting the RepAgent

These errors may occur when RepAgent starts up.

### Invalid Login

The RepAgent requires only a login name and password for the Replication Server user. The user must have **connect source** permission in the Replication Server.

RepAgent retries the login and error 9216 is reportedin the Adaptive Server log. If RepAgent continues to fail to log in to the Replication Server, RepAgent error 9214 is reported in the Adaptive Server error log.

### Invalid Permissions

The RepAgent user must have **connect source** permission for the Replication Server.

RepAgent error 9211 appears in the Adaptive Server error log if the user permission is invalid:

```
00:00000:00024:2006/06/23 14:44:45.12 server
RepAgent(4): Received the following error message from
the Replication Server: Msg 37024. CONNECT SOURCE
permission is required to execute command..
00:00000:00024:2006/06/23 14:44:45.24 server  Error:
9261, Severity: 20, State: 0
00:00000:00024:2006/06/23 14:44:45.27 server
RepAgent(4): This Rep Agent Thread is aborting due to
an unrecoverable communications or Replication Server
```

```
error.
00:00000:00024:2006/06/23 14:44:45.27 server  Rep Agent
Thread for database 'westRS_RSSD' (dbid = 4) terminated
abnormally with error. (major 92, minor 61)
```

# Errors from the Replication Server

Different types of problems can originate from the Replication Server.

Most Replication Server normalization errors, which result from inconsistencies in the setup of replication objects, are recoverable. The RepAgent logs the error and continues processing. Normalization errors are identified by error numbers 32000 – 32999.

Errors may still occur even when a replication definition is created and a subscription is successfully materialized. For example, an update may be lost, causing the replicate data to be inconsistent with the primary data. This kind of error should occur only during the initial setup of a replication system. The system administrator should monitor the error logs closely and correct errors as they are identified. Normalization errors may also occur with replicated stored procedures.

All other Replication Server errors, except network failures, are treated as fatal by the RepAgent. For example, an incorrect Replication Server login name and password in the configuration file is a fatal error. The RepAgent disconnects from Replication Server and terminates. After you have corrected the problem, restart the RepAgent.

### See also

- *Subscription Problems* on page 75

## Error 32032

Table is not defined in a database.

*Symptom*

Replication Server error 32032 is reported in the Replication Server error log:

```
Message: 32032, 'No table with name 'foo' is defined for
database with id MY_PDS.MY_PDB1.
```

*Explanation*

A table has been marked replicated with **sp_setreplicate**, but the replication definition has not yet been created for the table. The RepAgent retrieves log records for an object that is not yet known to the Replication Server.

*Solution*

Create the replication definition for the table.

## Error 32044

Incorrect definition of LOB column datatypes in the replication definition.

### *Symptom*

Replication Server error 32044 is reported in the Replication Server error log:

```
00:0003:00000:00031:2013/01/10 00:17:43.20 server
RepAgent(4): Error in passthru packet: 'distribute 4
0x000100000000256400003ddd000500003ddd00020000a1420004dd9400000000,
6
0x000100000000256400027265706c325f323434333874646232
 applied owner =~"$dbo ~"%tbl3.~!-rs_writetext append first last
changed with log textlen =2 ~$-unitext_fld1=~.!!#aa'.

00:0003:00000:00031:2013/01/10 00:17:43.20 server
RepAgent(4): Received the following error message from the
Replication Server: Msg 32044.
The datatype ('4') for 'tbl3.unitext_fld1' is not convertible to the
required type ('5').
```

### *Explanation*

In the last line of the error message: `The datatype ('4') for 'tbl3.unitext_fld1' is not convertible to the required type ('5').`, datatype 4 is `text` which is not convertible to datatype 5 which is `image`.

If you incorrectly define the column datatypes for the LOB columns in the replication definition, subscription materialization works but RepAgent cannot process data replication and therefore shuts down.

The datatype for `text`, `unitext`, and `image` columns in the Adaptive Server database is carried in the modification Log Transfer Language (LTL) commands that the RepAgent sends to the Replication Server. If the these datatypes do not match in both the Adaptive Server database and the replication definition, Replication Server detects the inconsistency when the modification is being replicated, and the RepAgent shuts down. The Replication Server records a warning message for the RepAgent in the Adaptive Server error log.

### Solution

When RepAgent shuts down because you did not define a large object (LOB) datatype column such a `image`, `text`, or `unitext` column correctly in the replication definition, you must purge corrupt data from affected queues and correct the replication definition before resuming replication.

This scenario creates table `t3` and the corresponding replication that incorrectly defines the datatypes in the table resulting in error 32044 and the shut down of RepAgent. The scenario shows how to purge the queues and recover from the error.

1. Create the `tbl3` table in the primary and replicate Adaptive Server databases:

```
create table tbl3
(
p_key int not null,
char_col char(10),
unitext_fld1 text null,
unitext_fld2 text null
)
```

**Remember:** The datatype of columns `unitext_fld1` and `unitext_fld2` in the schema for `tbl3` is `text` and the columns are not compressed LOB data columns.

2. Create the **repdef_tbl3** replication definition for the `tbl3` table in the `pdb1` primary database of the PDS data server:

```
create replication definition repdef_tbl3
with primary at PDS.pdb1
( p_key int,
  char_col char(10),
  unitext_fld1 image,
  unitext_fld2 image
 )
primary key (p_key)
searchable columns (p_key)
always_replicate (unitext_fld1,unitext_fld2)
 go
```

**Attention:** Instead of `text`, the replication definition is incorrect and defines the datatypes of columns `unitext_fld1` and `unitext_fld2` as `image`. The two columns are not compressed LOB data columns and therefore the replication definition does not need to define the columns as `image` datatype columns.

3. Insert data into `tbl3` at `pdb1`:

```
insert into tbl3 values (5,'cc','aa','bb')
go
```

The RepAgent shuts down because of the colmn datatype mismatch between the table schema and the replication definition and you see error messages in the log files of the :

- Primary Replication Server –

```
E. 2013/01/23 18:38:44. ERROR #32044 REP AGENT(PDS.pdb1) - /
nrm/nrm.c(11165)
        The datatype ('4') for 'tbl3.unitext_fld1' is not
convertible to the required type ('5').
```

- Primary Adaptive Server–

```
00:0003:00000:00037:2013/01/23 18:38:44.15 server  RepAgent(4):
Error in passthru packet:
'distribute 4
0x000000000000234b00003dbe000600003dbe00030000a14f013343db0000
0000,
6 0x000000000000234b00037265706c325f313233363874646232 applied
owner =~"$dbo ~"%tbl3.
~!-rs_writetext append first last changed with log textlen =2 ~
```

```
$-unitext_fld1=~.!!#aa '.
00:0003:00000:00037:2013/01/23 18:38:44.15 server
RepAgent(4): Received the following error message from the
Replication Server:
Msg 32044. The datatype ('4') for 'tbl3.unitext_fld1' is not
convertible to the required type ('5')..
```

4. Obtain the inbound queue number for `pdb1` in PDS. At Replication Server enter:

```
admin who
```

You see:

```
29 SQM         Awaiting Message     102:1 PDS.pdb1
27 SQM         Awaiting Message     102:0 PDS.pdb1
   REP AGENT   Down                 PDS.pdb1
```

The inbound queue number is 102.

5. Dump the contents of queue 102 to the client machine where you are issuing the **sysadmin dump_queue** command. The first open transaction after RepAgent shuts down contains the corrupt data. At Replication Server enter:

```
sysadmin dump_queue,102,1,-1,-1,-1,client
go
```

In the Replication Server log, you see:

```
102          1          0         15          0        240        102
        Jan 23 2013  6:38PM
0x000000000000234b00003dbe000300003dbe00030000a14f0133
       43db0000000000000000 sa       _ins

0x00000000000000000000000000000000000000000000000000000000000000000000
0000f0000
        4
0x000000000000234b00037265706c325f313233363874646232000000660000000000000

0000000000000000000000000000000000000000000000000000000000000000000000000
000000000

0000000000000000000000000000000000000000000000000000000000000000000000000
000000000
        00000000000000000000
        0        1100          19           0
        begin
transaction



       102          1          0         15          1
484         102
        Jan  1 1900 12:00AM

0x000000000000234b00003dbe000400003dbe00030000a14f013343db0000000000000000
        NULL         NULL
```

```
0x00000000000000000000000000000000000000000000000000000000000000000000
0000f0001
        2101248

0x00000000000234b00037265706c325f313233363874646232000000660000000000000000

0000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000000000000000000000000000000
        00000000000000000000
        0        1100         104           0
        insert into dbo.tbl3 (p_key, char_col, unitext_fld1,
unitext_fld2)
        values (5, 'cc        ', 0x20, 0x20)
Note this transaction is interrupted. only begin transaction and
insert.
```

6. Purge the first open transaction from the 102 inbound queue. At Replication Server enter:

```
sysadmin purge_first_open,102,1
go
```

7. Dump the queue contents again to check if there are any error messages due to the continued presence of corrupt data in the queues:

```
sysadmin dump_queue,102,1,-1,-1,-1,client
go
```

8. Purge the queue until there are no error 32044 messages in the dump from the queue.

```
sysadmin purge_first_open,102,1
go
```

9. Correct the replication definition by changing the datatypes of the `unitext_fld1` and `unitext_fld2` columns to `text` to match the table schema:

```
alter replication definition tbl3
alter columns with unitext_fld1 text
go
alter replication definition tbl3
alter columns with unitext_fld2 text
go
```

10. Enable autocorrection for the DSI thread:

```
set autocorrection on
for replication_definition
with replicate at data_server.database
```

11. Retrieve the current database generation ID.
    At the Adaptive Server primary database, enter:

```
dbcc gettrunc
go
```

    You see a single row

```
secondary trunc page secondary trunc state dbrepstat generation
id
------------------- -------------------- ---------
-------------
               15836                    1       231             1

database id database name ltl version
----------- ------------- -----------
          4 pdb1          760
```

12. Instruct Adaptive Server to increment the database generation number in the log for the database where RepAgent has shut down
    Since the value for the "generation id" column output from **dbcc gettrunc** is 1, set the new ID value to 2:

    ```
    dbcc settrunc('ltm', 'gen_id', 2)
    go
    ```

13. Restart RepAgent
14. Disable autocorrection after successful replication of the data that was previously corrupted:

    ```
    set autocorrection off
    for replication_definition
    with replicate at data_server.database
    ```

## Error 32046

Inconsistencies in the replication status of `text`, `unitext`, or `image` columns between the Adaptive Server database and the replication definition.

### *Symptom*

Replication Server error 32046 is reported in the Adaptive Server error log (for RepAgent):

```
Message: 32046 -- 'The status of column '%s' in repdef
is inconsistent with that of the LTL command.'
```

### *Explanation*

The replication status for `text`, `unitext`, and `image` columns in the Adaptive Server database is carried in the modification Log Transfer Language (LTL) commands that the RepAgent sends to the Replication Server. If the status of a `text`, `unitext`, or `image` column is not the same in both the Adaptive Server database and the replication definition, Replication Server detects the inconsistency when the modification is being replicated, and the RepAgent shuts down.

If a `text`, `unitext`, or `image` column has a status of **do_not_replicate** at the Adaptive Server database and the replication definition includes that column for replication, processing continues and the Replication Server sends the modifications to the replicate database without the `text`, `unitext`, or `image` data. The Replication Server records a warning message in the Adaptive Server error log (for the RepAgent).

*Solution 1*

When the RepAgent shuts down because a `text` or `image` column has a status of **replicate_if_changed** at the Adaptive Server database and **always_replicate** in the replication definition, you must change the replication status so that they match.

To replicate `text`, `unitext`, or `image` columns only when their values change:

1. Execute the **alter replication definition** command at the primary Replication Server and change the status of the `text`, `unitext`, or `image` columns to **replicate_if_changed**. Wait for the modified replication definition to arrive at the replicate sites.
2. Restart the RepAgent.

To always replicate `text`, `unitext`, or `image` columns:

1. Stop updates at the primary table.
2. Execute the **alter replication definition** command at the primary Replication Server, and change the status of the `text`, `unitext`, or `image` columns to **replicate_if_changed**. Wait for the modified replication definition to arrive at the replicate sites.
3. Restart the RepAgent to let transactions with a **replicate_if_changed** status finish processing.
4. Execute **sp_setrepcol** at the Adaptive Server and change the status to **always_replicate**.
5. Execute **alter replication definition** at the primary Replication Server and change the status of the `text`, `unitext`, or `image` columns to **always_replicate**. Wait for the modified replication definition to be replicated to the replicate sites.
6. Resume updates to the primary table.

*Solution 2*

When the Replication Server reports that the status of a `text` or `image` column is **do_not_replicate** at the Adaptive Server database and the replication definition includes that column for replication, you must change the replication status to either:

- Replicate `text`, `unitext`, or `image` columns, or
- Not replicate `text`, `unitext`, or `image` columns.

To replicate `text`, `unitext`, or `image` columns:

1. Execute **sp_setrepcol** at the Adaptive Server database and change the status of the `text`, `unitext`, or `image` column to **always_replicate** or **replicate_if_changed**. It should match the status in the replication definition.
2. Wait for subsequent transactions that modify the `text`, `unitext`, or `image` column to be processed by the Replication Server.
3. Consider correcting any inconsistencies with the **rs_subcmp** program.

To not replicate `text`, `unitext`, or `image` columns:

1. Stop updates to the primary table.
2. Drop subscriptions to the replication definition.
3. Drop the replication definition.
4. Re-create the replication definition without the `text`, `unitext`, or `image` columns, and re-create subscriptions.
5. Resume updates to the primary table.

### Error 32047

A stored procedure declared for function replication is marked for table replication.

*Symptom*

Replication Server error 32047 is reported in the Replication Server error log:

```
Message from server: Message: 32047, State 0, Severity
12 -- 'Function 't2' is associated with a function
replication definition, but an invoking stored
procedure was marked for a table replication
definition.'.
```

*Explanation*

The replicated stored procedure is declared in the Replication Server for function replication but is marked in the Adaptive Server for table replication.

*Solution*

Correct replication definition setup. See *Replication Server Administration Guide Volume 1*.

## Errors from the Adaptive Server

Errors from the Adaptive Server can be a result of an invalid truncation page or of running out of Adaptive Server alarm resources. The RepAgent does not use the Adaptive Server log scan thread, and will not encounter any errors related to it.

### Possible Issues when Dropping Primary Objects

Drop primary tables or stored procedures from an Adaptive Server with caution because doing so affects data replication.

If updates are made to the primary table and the table is then dropped from the Adaptive Server, Adaptive Server error 9104 is displayed in the Adaptive Server error log (for the RepAgent) :

```
Message: 9104 'Cannot identify the object on the INSERT
log record for database 'MY_PDB1', XACT ID Rid pageid =
0xa0f; row num = 0x5, RID Rid pageid = 0xa0f; row num =
0x6. Information associated with the INSERT log record
is not replicated.
```

The log record for a replicated data operation references another log page, called the `oampage`, that has information about the replicated object. Since the table was dropped, the `oampage` is deallocated and the data cannot be replicated. If the `oampage` is assigned to a newly created object, the Adaptive Server may associate log records for the dropped object with the new object.

In this example, the `oampage` for `foo` may be allocated to `foo_bar` after `foo` is dropped:

```
update table foo
 drop table foo
create table foo_bar
 update foo_bar
```

Adaptive Server sends the first update record as an update to `foo_bar` instead of `foo`. If the new table has a much larger row size, then an Adaptive Server segmentation fault may occur.

Dropping or re-creating replicated stored procedures leads to unpredictable results; schema alterations have the same effect. Make these type of changes only after the RepAgent has processed all the records for the stored procedure.

Also, check the Replication Server for error messages.

## Invalid Truncation Page

The secondary truncation point identifies the page that contains the **begin transaction** statement of the most recent transaction that was sent to the Replication Server.

The RepAgent executes **dbcc settrunc** on the Adaptive Server to set the secondary truncation point in the database log. If the RepAgent specifies an invalid page to the Adaptive Server, an error is recorded and the RepAgent aborts. You may see Adaptive Server error 4213:

```
Message: 4213, State: 1, Severity: 16 -- 'Page 2561 in
database 'MY_DB' cannot be used as a truncation page
because the page is not allocated.'
```

The `rs_locater` table in the Replication Server system database tells the RepAgent where to set the secondary truncation point in the Adaptive Server log. This error is expected if the truncation state is set to **ignore** and the log is truncated. The RepAgent then restarts without resetting `rs_locater`.

To solve this problem, set `rs_locater` to zero and restart the RepAgent.

### See also

- *Resetting the Database Log Locater* on page 130

# Data Server Interface Problems

The Data Server Interface (DSI) applies transactions from a Replication Server stable queue to a data server, which can be an Adaptive Server or another data server with an interface to the Replication Server. Before a transaction is applied, Replication Server uses function strings to convert the commands to the format specified for the data server. If the data server does not receive updates, the DSI may be suspended or down.

A down or suspended DSI can be caused by:

- Incorrect login or permissions
- A data server error
- A Replication Server error

A suspended DSI always results in a message being sent to the Replication Server error log. Analyze the message to solve the problem. After correcting the error, restart the DSI.

If the DSI is active, unique transactions might be incorrectly resolved as duplicates, or the transactions might be failing on the data server. Manually resolve the data inconsistency between the primary and replicate databases or reapply the transactions.

To investigate the DSI, find out which databases are controlled by the Replication Server and check the DSI status for all database connections on a Replication Server. Use the **select**, **admin who**, and **admin who, dsi** commands to extract the information you need and to determine the exact cause of the problem.

**Note:** These problems also apply to parallel DSI threads. For more information specific about parallel DSI threads, see *Replication Server Administration Guide Volume 2*.

### See also

- *Common Error Messages* on page 35
- *Troubleshooting Overview* on page 5

## Listing Databases Controlled by a Replication Server

Identify the databases controlled by a Replication Server.

1. Log in to the system Adaptive Server.
2. Change to the Replication Server System Database (RSSD).
3. Enter:

```
select dsname, dbname from rs_databases, rs_sites
 where prsid = id and name = replication_server
```

# admin who and admin who, dsi

**admin who** and **admin who, dsi** checks the Data Server Interface (DSI) status for a database connection.

The command **admin who** displays the states of all threads in the Replication Server, including DSI scheduler and executor threads. In **admin who** output, DSI scheduler threads are identified as "DSI" and DSI executor threads are "DSI EXEC." If the DSI is suspended when Replication Server starts, the output shows only one DSI executor thread, even if additional DSI executor threads are configured.

**admin who, dsi** shows the states of all running DSI scheduler threads and provides configuration values and other information about them. If a DSI scheduler thread exists for a database but does not appear in the **admin who, dsi** output, use the **resume connection** command to restart the DSI for the database. For a complete description of **admin who, dsi** output, see the *Replication Server Reference Manual*.

## States of the DSI Scheduler Thread

Explains the Data Server Interface (DSI) scheduler thread states used in the **admin who** and **admin who, dsi** output.

- Active – the thread is starting, restarting after an internal error, or logging an exception to the RSSD.
- Awaiting Command – the thread is waiting for a transaction to become available in the stable queue for the database. There are no complete transactions in the queue.
- Awaiting Wakeup – the thread sleeps for two minutes after an error that can be retried. During the two-minute interval, the cause of the error may disappear or the system administrator may correct the problem. If the problem is corrected during the sleep interval, the thread restarts with no errors.

  A failed transaction is retried when it causes a data server error that, using **assign action**, have been assigned a **retry_stop** or **retry_log** error action.
- Awaiting Message – the thread has dispatched transactions to the DSI executor threads and is waiting for them to complete.
- Suspended – the DSI connection has been suspended by a user command, an error, or a **drop subscription** or **activate subscription** command with the **with suspension** clause. This state appears only in **admin who** output.
- Down – the thread has not been started. This state appears only in **admin who** output.

## States of the DSI Executor Thread

Explains the Data Server Interface (DSI) executor thread states used in the **admin who** output.

- Active – the thread is starting, executing a transaction at the data server, logging an exception to the RSSD, or restarting after an internal error.

- Awaiting Message – the thread is processing a transaction and is waiting for another thread to complete processing its transaction, or the thread is waiting to receive another statement from the Stable Queue Transaction interface (SQT).
- Awaiting Command – the thread is waiting to receive another transaction from the DSI scheduler thread.
- Down – depends on the state of the DSI scheduler thread:
  - If the DSI scheduler thread is Down, the DSI executor thread connection was suspended when the Replication Server was started, and the connection has not been resumed.
  - If the DSI scheduler thread is Active or Awaiting Wakeup, the DSI executor thread connection is recovering from a retryable error and is starting or restarting.
- Suspended – the connection has been suspended by a user command, an error, or a **drop subscription** or **activate subscription** command using the **with suspension** clause.

  A failed transaction does not cause the DSI connection to be suspended and is retried if the failed transaction caused a data server error that, using **assign action**, have been assigned the **retry_stop** or **retry_log** error action. For more information about the command, see *Replication Server Reference Manual > Replication Server Commands > assign action*.

## Troubleshooting the DSI for the Replicate Database

The Data Server Interface (DSI) for the replicate database may have been suspended if changes made to a primary database do not reach the replicate database that has active subscriptions.

Use this procedure to find out the cause of the problem.

1. Log in to the Replication Server that controls the replicate database.
2. Execute **admin who, dsi**.

   This returns one entry for each database with replicated data.

   - If a database does not have an entry, check the Replication Server error log to see if the DSI for the database was not started or was suspended. These are samples of error messages that are reported in the Replication Server error log as a result of a DSI that did not start or was suspended:

     ```
     The DSI thread for 'RDS.rdb2' is not started, because
     the connection to the database has been suspended or
     the connection has not been completed.
     ```

     ```
     The DSI thread for database 'RDS.rdb2' is being
     shutdown. DSI received data server error #1105 which
     is mapped to STOP_REPLICATION. See logged data
     server errors for more information. The data server
     error was caused by RS output command #1 mapped from
     input command #1 of the failed transaction.
     ```

     ```
     The DSI thread for database 'RDS.rdb2' is shutdown.
     ```

- If there is an entry for the database, but the state is Awaiting Command, the DSI is waiting for a transaction from the primary. Determine why updates are not reaching this Replication Server.
- If the state of the DSI thread is Active, then:
  - Unique transactions might be incorrectly resolved as duplicates, or
  - Transactions might be failing on the data server and written to the exceptions log.

**See also**

- *Errors When DSI is Down or Suspended* on page 120
- *Errors When DSI is Active* on page 123
- *admin who and admin who, dsi* on page 118
- *Troubleshooting Replication Failures* on page 21

# Errors When DSI is Down or Suspended

Incorrect login or permissions, data server errors, or Replication Server errors can cause the Data Server Interface (DSI) to suspend or shut down.

If a subscription problem causes the DSI thread for the replicate database to terminate abnormally, you can restart the thread using the **resume connection** command. If possible, fix whatever condition caused the problem before resuming the connection. For example, if the maintenance user does not have **update** permission on the replicate table, first grant the user **update** permission, then resume the connection.

If you cannot fix the problem, resuming the connection causes the DSI thread to reexecute the command that failed, and suspends the DSI again. To prevent this cycle, assign a different action to the error returned to the DSI. You must assign error actions at the Replication Server where the error class is created. For information about error actions and classes, see *Replication Server Administration Guide Volume 2*.

## Connection Failure to the Database

Troubleshoot connection failures.

If the error is a connection failure to the database, verify that the:

- Data server is defined in the `interfaces` file.
- Data server is running.
- Maintenance user name and password are correct.

To find the maintenance user name and password for a database, log in to the system Adaptive Server, and execute this query using the Replication Server System Database (RSSD):

```
select username, password
     from rs_maintusers, rs_databases
     where destid = dbid
```

```
    and dsname = data_server
   and dbname = database
```

If you have password encryption, you cannot access the password by executing a query. If the maintenance user name or password is incorrect, change it on the data server or use **alter connection** to change it in Replication Server. After correcting the user name or password, resume the connection.

### *Asynchronous Transaction Connection Failures*

When an asynchronous stored procedure reaches the primary Replication Server, the Data Server Interface (DSI) uses the original login and password to connect to the primary data server. The maintenance login is not used.

If the login fails, use **sysadmin log_first_tran** to log the asynchronous transaction into the exceptions log. The `app_user` and `app_pwd` columns in the `rs_exceptshdr` system table contain the login and password the DSI uses to log in to the primary data server.

#### See also

- *Examining the Exceptions Log* on page 124

## Data Server Errors

Data server errors, such as permission violations or duplicate keys, are logged in the Replication Server error log.

If you cannot correct the data server error, restart the Data Server Interface (DSI), requesting that the first transaction (the one causing the error) be skipped. Execute **resume connection** with the **skip transaction** option.

**Warning!** Skipping transactions may cause replication inconsistencies.

If errors are caused by unique-key violations, use **set autocorrection** to turn automatic error correction on, before you resume the connection. These errors are likely to occur during nonatomic materialization. For more information about **set autocorrection**, see the *Replication Server Reference Manual*.

If using **set autocorrection** does not work, use the **skip transaction** option of the **resume connection** command. This option skips the materializing or dematerializing transaction altogether. If the problem occurred during subscription materialization, drop the subscription and create it again in a way that avoids the same problem.

#### See also

- *Subscription Problems* on page 75

### Implications of Skipping Transactions

Skipping transactions that cause errors forces the Data Server Interface (DSI) to resume applying transactions to the database. However, this method has important implications for the database.

A transaction intended for a database is skipped when:

- You execute **resume connection** with the **skip transaction** option.
- A data server error is encountered and the assigned action for the error is **log**, or **retry_log**.

Skipped transactions carry these implications:

- Replicate data may become inconsistent with primary data.
- If an asynchronous transaction originates in a replicate database, and the transaction is skipped at the primary data server, the primary database is not updated.
- The inconsistency caused by skipping a transaction may result in additional errors if subsequent transactions depend on the unapplied actions of the skipped transaction.
- Subscription materialization and dematerialization requests are special types of transactions. If you skip a subscription materialization request, it may invalidate the subscription and must be dropped.
- A subscription materialization or dematerialization request may create a separate stable queue, called the materialization or dematerialization queue. You cannot use **resume connection** to explicitly skip transactions in this queue. However, transactions may be skipped due to data server errors that are assigned the **log** or **retry_log** action.
- If a subscription materialization or dematerialization request is skipped, all transactions in the corresponding queue may be skipped. However, if a DSI is suspended in the middle of applying the transactions in a materialization or dematerialization queue, some of the transactions in the queue may already have been committed while others may not. Later, if you use **resume connection** to skip the request, the replicate database has the effects of the previously committed transactions, where some of the transactions in the queue were applied to the replicate database although the materialization or dematerialization request was skipped after the failure.

Because of these implications, skip transactions only after exhausting other means of correcting the error. After skipping a transaction, determine what you must do to bring the replicated data back to a state of consistency.

### Customized Handling of Data Server Errors

Replication Server can customize its response to data server errors.

If you want the Data Server Interface (DSI) to continue applying updates even when a data server error is encountered, use **assign action** at the Replication Server where the error class was created to change the error action from **stop_replication** to **log** or **retry_log**. All databases using the error class are affected. If the primary database for a class is at a different site, you may need to wait until the new error action is replicated to the local site.

To limit the change to one database, create a new error class with the **create error class** command. Specify the desired error actions for the class, then change the connection to the database to use the new error class.

To examine the commands in the transaction that caused the connection to be suspended, use **sysadmin log_first_tran** to write the first transaction into the exceptions log without skipping it.

## Replication Server Errors

Troubleshoot Replication Server errors recorded in the Replication Server error log.

| Problem | Suggested Action |
|---|---|
| Function strings | See:<br>• *Replication Server Troubleshooting Guide > Troubleshooting Overview > Types of Replication System Problems > Subscription Problems > Function String Restrictions*<br>• *Replication Server Troubleshooting Guide > Common Error Messages > Replication Server Error Messages > 29024* |
| Subscriptions | See *Replication Server Troubleshooting Guide > Subscription Problems*. |
| Missing objects | This is probably caused by some inconsistency in the Replication Server System Database (RSSD). Call Sybase Technical Support for assistance. |

### See also

- *Subscription Problems* on page 75
- *Function String Restrictions* on page 15
- *Error 29024* on page 53

# Errors When DSI is Active

When the Data Server Interface (DSI) is active but the replicate data server is not receiving updates, unique transactions might be incorrectly resolved as duplicates, duplicate transactions might be reapplied, or transactions might be failing on the data server.

If the error's action is set to **log** or **retry_log**, error and failed transaction are written to the exceptions log. Manually resolve the data inconsistency between the primary and replicate databases or reapply the transactions.

## Incorrect Duplicate Transaction Resolution

If the `origin_qid` values stored in a data server or the `rs_exceptslast` table is modified by mistake, non-duplicate transactions may be ignored, or duplicate transactions may be reapplied.

If you suspect that this is happening, check the stored values and compare them with the transactions in the database's stable queue. If the values are incorrect, modify them directly.

The Data Server Interface (DSI) records the most recent transaction committed or written into the exceptions log so it can detect duplicates after a system restart. Each transaction is identified by a unique origin database ID and an origin queue ID that increases for each transaction.

The last transaction committed from each origin database is recorded on a data server through execution of the function strings defined for the data server's function-string class. For the default function-string class, `rs_sqlserver_function_class`, this is done in the function string of a **commit** command, that is, the **rs_commit** function. The function-string class of every data server must support the **rs_get_lastcommit** function, which returns the `origin_qid` and `secondary_qid` for each origin database. The `secondary_qid` is the ID of the queue used for subscription materialization or dematerialization.

The `origin_qid` and `secondary_qid` for the last transaction written into the exception log from each origin is recorded in the `rs_exceptslast` table in the Replication Server System Database (RSSD). However, transactions logged explicitly by the **sysadmin log_first_tran** command are not recorded in this table. These transactions are logged, but they are not skipped.

When a data server interface is started or restarted, it gets the `origin_qid` returned by the **rs_get_lastcommit** function and the one stored in the `rs_exceptslast` table. It assumes that any transaction in the queue with an `origin_qid` less than the larger of these two values is a duplicate and ignores it.

### See also

- *Stable Queues* on page 139

## Examining the Exceptions Log

Examine the exceptions log to determine the transactions that have been skipped.

Skipped transactions are written into the exceptions log. Orphan transactions and transactions logged by **sysadmin log_first_tran** are also in the exceptions log.

If a replicate database is not receiving updates, the update transactions may have been skipped and written into the exceptions log.

The exception log consists of three tables: `rs_exceptshdr`, `rs_exceptscmd`, and `rs_systext`. The `rs_exceptshdr` table has one entry for each transaction. The

`rs_exceptscmd` table has one entry for each command (either source or output) of the transaction. The `rs_systext` table stores the text of the commands. See the *Replication Server Reference Manual*.

1. Log in to the system Adaptive Server for the Replication Server controlling the database.
2. Execute this query against the Replication Server System Database (RSSD) to view the header information for all logged transactions intended for a database:

   ```
   select * from rs_exceptshdr
        where error_site = data_server
       and error_db = database
       order by log_time
   ```

   The rows are returned in ascending order, of the time the transactions were logged. To list the rows in descending order, include “desc” at the end of the query.
3. Join the three system tables to view all information about a transaction in the log.

   This query gives the source commands and their corresponding output commands for each logged transaction:

   ```
   select hdr.sys_trans_id, cmd_type, textval
        from rs_exceptshdr hdr,
            rs_exceptscmd cmd,
            rs_systext
        where error_site = data_server
          and error_db = database
           and hdr.sys_trans_id = cmd.sys_trans_id
            and cmd_id = parentid
        order by log_time, src_cmd_line,
            output_cmd_index, sequence
   ```

# Adaptive Server Log Problems

The Adaptive Server primary database log is the source of the data that Replication Server distributes. A RepAgent retrieves transactions from the log and sends them to the Replication Server.

Adaptive Server log problems include:

- Log files that have reached the maximum size and need to be truncated
- Missing or incomplete transactions at the primary Replication Server as a result of truncated primary database logs
- Log files corrupted by software and hardware sources

### See also

- *Common Error Messages* on page 35
- *Troubleshooting Overview* on page 5

## Truncating an Adaptive Server Log

Truncate the database log when it is too full to allow Adaptive Server to continue processing updates.

Adaptive Server uses truncation points to ensure that only transactions processed by the RepAgent are truncated. A secondary truncation point marks the place in the primary database log up to which the RepAgent has processed transactions. The RepAgent periodically updates the secondary truncation point to reflect transactions successfully passed to the Replication Server. Adaptive Server does not truncate the log past the secondary truncation point. See *Replication Server Administration Guide Volume 1*.

At times you may need to truncate the log beyond the secondary truncation point. For example, if the RepAgent cannot access the Replication Server and the log fills, you may want to truncate the log at the secondary truncation point rather than extending the log or preventing clients from updating the primary database. Truncating the log can cause inconsistencies between the primary and replicate databases.

1. Verify that the secondary truncation point is set for the database.
2. Turn off the secondary truncation point in the database.
3. Dump the database log. See the *Adaptive Server Enterprise Reference Manual*.
4. Set the secondary truncation point.

### See also

- *Log Truncation Problems* on page 130

## Verifying the State of the Secondary Truncation Point

Check whether the secondary truncation point is set for a database.

1. Log in to the primary database.
2. Execute **dbcc gettrunc**.
   For example, to see if the secondary truncation point is set in the `Parts` database:

   ```
   > use Parts
   > go
   > dbcc gettrunc
   > go
   ```

   In the output, a 0 in the `ltm_trunc_state` column means the secondary truncation point is turned off for the database. A 1 in the `ltm_trunc_state` column indicates the secondary truncation point is turned on.

## Turning Off the Secondary Truncation Point in a Database

Disable the secondary truncation point for a database to truncate portions of the log that have not been transferred. Truncating the log frees only the log pages that the RepAgent has sent to the Replication Server with confirmation.

**Note:** Only the Adaptive Server system administrator and the database owner have permission to execute the **dbcc settrunc** command.

1. Suspend the RepAgent that is forwarding the log.

   Otherwise, executing **dbcc settrunc** results in the following error:

   ```
   The log transfer context for the current database is
   already reserved by Adaptive Server process 7. The log
   transfer context for the current database is not
   reserved.
   ```

2. Log in to the Adaptive Server that contains the database for which to turn off the secondary truncation point.
3. Execute **dbcc settrunc(ltm, ignore)** to turn off the secondary truncation point.
   To turn off the secondary truncation point in the database named `Parts`:

   ```
   > use Parts
   > go
   > dbcc settrunc(ltm, ignore)
   > go
   ```

4. Truncate the log using the Transact-SQL **dump transaction** command. See *Adaptive Server Enterprise Reference Manual*.
5. Dump the transaction log.

   This may result in the loss of updates to the replicate database. The primary database is not affected. You must reapply the lost updates to the replicate database. See *Replication Server Administration Guide Volume 2*.

### See also

- *Log Truncation Problems* on page 130

## Setting the Secondary Truncation Point

Reestablish the secondary truncation before restarting a RepAgent.

1. Log in to the Adaptive Server that contains the database for which to reestablish the secondary truncation point.
2. Execute **dbcc settrunc(ltm,valid)** to reestablish the secondary truncation point.
   For example, to reestablish the secondary truncation point in the `Parts` database:

   ```
   > use Parts
   > go
   > dbcc settrunc(ltm, valid)
   > go
   ```

3. Verify that the secondary truncation point has been enabled:

   ```
   > use Parts
   > go
   > dbcc gettrunc
   > go
   ```

   The output should show 1 in the `ltm_trunc_state` column.
4. Reset the locater for the database in the `rs_locater` table.
5. Resume the Replication Server to restart the RepAgent.

The RepAgent begins scanning the log from the page returned in the `ltm_truncpage` column by the **dbcc gettrunc** command.

# Database Log Locater

When a RepAgent starts, it uses a *locater* to determine where to begin scanning the database log. The locater for a database log is stored in the `rs_locater` table in the Replication Server System Database (RSSD) of the Replication Server that controls the database.

**Note:** This locater is different from the Replication Server Interface (RSI) locater.

The locater points to a record on a page in the database log. If the truncation point for the database is turned off, you or Adaptive Server may truncate the page the locater points to, invalidating the locater. When the locater is invalid, the RepAgent writes error message 9215 to the Adaptive Server error log.

To recover, reset the locater to 0x0. This directs the RepAgent to begin scanning the log from the truncation point.

### See also

- *Invalid RSI Locater* on page 105

- *Error 9215 (ASE 624)* on page 63

## Resetting the Database Log Locater

Reset the locater to 0x0 to direct RepAgent to begin scanning the log from the truncation point.

Update the `rs_locater` table in the Replication Server System Database (RSSD) for the Replication Server controlling the database using the Adaptive Server stored procedure **rs_zeroltm**:

```
rs_zeroltm data_server, database
```

where *data_server* and *database* are the respective Adaptive Server and database for which to set the locater value.

For example, to reset the locater for the MYDS Adaptive Server and the `Parts` database, execute:

```
rs_zeroltm MYDS, Parts
```

# Log Truncation Problems

Inconsistencies may occur when the primary database log is truncated.

- Orphaned transactions can cause the stable queues to fill.
- Inbound queue requires more disk space.
- Parts of transactions, or entire transactions, may be missing, causing inconsistent data at replicate sites. See *Replication Server Administration Guide Volume 2*.
- Subscription commands may be lost, disrupting the materialization or dematerialization protocol and requiring manual cleanup.
- If a log is truncated while a route is being created, subscription commands against the Replication Server System Database (RSSD) may be lost. As a result, the route cannot be created. If this occurs, drop and re-create the route.

For more information about how to recover lost transactions, see *Replication Server Administration Guide Volume 2*.

## Checking for Orphaned Transactions

Commit or roll back orphaned transactions to avoid filling up the inbound queue.

An orphaned transaction is a transaction in an inbound stable queue that is missing a terminating **commit** or **rollback** command. Because Replication Server does not free up a queue segment until every transaction in the queue segment (or in any of the preceding queue segments) has been committed or rolled back, orphaned transactions can cause the inbound queue to fill up.

1. Log in to the Replication Server.

2. Execute **admin who, sqt** to display information about the stable queues at the Replication Server.
3. Use the output from **admin who, sqt** to identify the entry for the inbound queue of the database whose log was truncated.

   The queue has a two-part name formed from the database ID and the queue-type identifier. For an inbound queue, the queue-type identifier is 1. For example, if the database ID is 101, the inbound queue name is 101:1.
4. Verify that an orphaned transaction exists.

   If the output for the inbound queue shows an open transaction that does not change over long periods of time, the queue probably contains an orphaned transaction. However, it may be difficult to distinguish between an orphaned transaction and a very long transaction.

   a) Dump the stable queue and examine the information about the transaction.

      You can dump only the `begin` record to find the user name and the time the command was executed, and then see if the user still has open transactions in the database.

   b) Dump the last block in the queue and look at the date for the commands. To determine whether this is the case, dump the last block in the queue and look at the date for the commands.

      You need to dump the last block because even if you find that the user does not have open transactions, the queue may still have an orphaned transaction. If the queue is large, the RepAgent may not be keeping up with it. Use **admin who, sqm** to find the last block.
5. Use **sysadmin purge_first_open** to skip an orphaned transaction.
6. Manually apply the orphaned transaction to the replicate database using the output from dumping the stable queue.

### See also

- *Stable Queues* on page 139

## Inbound Queue Requires More Disk Space

Problems such as orphaned transactions and connection failures force stable queues to hold more transactions than anticipated.

If the inbound queue requires more disk space, you see a message similar to this in the Replication Server error log:

```
_SQM_ADD_SEGMENT: Going to wait for a segment to be
freed. Administrative action is needed.
```

To determine which stable queues hold the segments, log in to the RSSD and execute:

```
select q_number, q_type, count(*)
 from rs_segments
 group by q_number, q_type
```

The output of this query shows the number of segments each stable queue is using. Each segment is 1MB of disk space. Look at the queues with the largest number of segments:

- A *q_type* of 0 means an outbound queue. If any outbound queues have more than two segments, verify that the connections they service are operating.
- A *q_type* of 1 is an inbound queue. If any of these queues has a large number of segments, it may contain an orphaned transaction.

### See also

- *Data Server Interface Problems* on page 117
- *Replication Server Interface Problems* on page 103
- *Checking for Orphaned Transactions* on page 130

## Symptoms of a Corrupted Adaptive Server Log

Adaptive Server log corruption can cause failures anywhere along the replication path.

- RepAgent shuts down when it reads log data it cannot process.
  The RepAgent shutting down is the most common symptom of a corrupt Adaptive Server log. In addition, the error log may reveal error messages that are inconsistent with the current operation; however, no specific log corruption error message is reported.
- A component of the Replication Server shuts down.
  When the RepAgent passes incorrect data to the Replication Server, Replication Server components, such as the Data Server Interface (DSI), may shut down. For instance, if the DSI encounters a mismatched data type and fails to resolve the differences, it shuts down.
- Replicated data appears out of sync.

The procedures for determining log corruption and where the corruption occurs is complex. It requires knowledge of the log format as well as Adaptive Server and Replication Server internal components.

**Note:** If you suspect Adaptive Server log corruption, contact Sybase Technical Support to help you diagnose and resolve the issue.

# Replication Manager Plug-in Problems

The Replication Manager (RM) plug-in provides a graphical user interface where you can manage, monitor, and troubleshoot most replication system components (including primary and replicate data servers, Replication Agents, and database gateway servers).

## Incorrect Case of Server Name in the sql.ini file

Using the incorrect case for the server name in `sql.ini` causes the RM plug-in to display odd results.

*Symptom*
Although the database is participating in replication, the connection folder is empty; replication definition and subscription folders do not appear.

*Explanation*
On UNIX, Open Client connectivity is case-sensitive; however, on Windows, it is not. The RM plug-in cannot ignore case when it is issuing commands.

*Action*
Verify that the server names in the `sql.ini` file used by the plug-in use the same case-sensitivity as the the server.

On the Replication Server RSSD, execute: To diagnose, do a on the Replication Server RSSD.

```
rs_helpdb
```

The exact name of the server and database are compared to that used in the RM plug-in to determine if a database participates in replication. If the server or database name does not match, the RM plug-in reports that the server/database is not not currently participating in replication from this Replication Server.

# Replication Monitoring Services Problems

The Replication Monitoring Services (RMS) is a middle-tier monitoring layer that monitors the status of the servers and components in the replication environment. It also provides information to troubleshoot problems and the commands to fix these problems.

The RMS is implemented using the Sybase Unified Agent Framework (UAF). The UAF provides flexible and extensible service-oriented approach to manage distributed resources. It provides common services and interfaces for development and deployment of agent plug-ins like RMS.

The RMS provides a tracing feature to assist the user when troubleshooting problems. When you use this feature, information about the internal workings of the RMS is written to the RMS log file. The user controls the type of information that is logged by setting trace flags in the RMS. For information on the RMS trace commands and a complete list of the trace flags, see the *Replication Server Reference Manual.*

### *UAF and RMS Error Logs*

Both the RMS and the UAF write errors, warnings, and informational messages to log files. Use these log files to troubleshoot issues in the RMS:

- UAF log file: `$SYBASE/UAF-2_5/log/agent.log`
- RMS log file: `$SYBASE/UAF-2_5/plugins/com.sybase.rms/logs/rms.log`

For more information on RMS, see the *Replication Server Administration Guide Volume 1.*

## List of RMS Trace Flags

Trace flags control the type of information that is recorded in the Replication Monitoring Services (RMS) log file.

**Table 21. Trace Flags for Troubleshooting RMS**

| Trace flags | Description |
|---|---|
| **trace RMS_Command on** | Writes to the error log every command the RMS receives. Commands can be sent from the RM plug-in, an isql session, or some other client application. This flag is maybe useful for identifying the command that caused a specific problem. |
| **trace Server_Command on** | Writes to the error log every command RMS sends to a monitored server. Use this flag to determine how RMS retrieves information about the replication environment. |

| Trace flags | Description |
| --- | --- |
| **trace Startup on** | Adds trace messages to the RMS log file at each step of the start-up process. Use this flag to diagnose Unified Agent Framework (UAF) start-up problems. |
| **trace Network_Connection on** | Adds trace messages to the RMS log file whenever a connection to a server is created. The RMS will include all connection information except the password in the trace message. Use this flag to trace network connection errors. |

See the *Replication Server Reference Manual* for a complete list of the trace flags.

## Multiple UAF Servers on the Same Computer

Executing multiple UAF servers on the same computer causes the Replication Monitoring Services (RMS) to report inconsistent and false server status.

*Symptom*

At UAF server start-up, the RMS generates this message if a UAF server is already running:

```
2006-03-10 12:57:16,520 INFO [main] RMIService(136) -
Starting RMI Service...
2006-03-10 12:57:16,629 ERROR [RMIRegistryThread]
RMIRegistryServiceStarter(34)- Failed to start rmi
registry. java.rmi.server.ExportException: Port already in use:
9999;
nested exception is: java.net.BindException: Address already in use
at
sun.rmi.transport.tcp.TCPTransport.listen
(UnknownSource) at
sun.rmi.transport.tcp.TCPTransport.exportObject
(Unknown Source) at
sun.rmi.transport.tcp.TCPEndpoint.exportObject
(Unknown Source) at
sun.rmi.transport.LiveRef.exportObject
(Unknown Source) at
sun.rmi.server.UnicastServerRef.exportObject
(Unknown Source) at
sun.rmi.registry.RegistryImpl.setup(Unknown Source) at
sun.rmi.registry.RegistryImpl.<init>
(Unknown Source) at
java.rmi.registry.LocateRegistry.createRegistry
(Unknown Source) at
com.sybase.ua.services.rmi.RMIRegistryServiceStarter.
run(RMIRegistryServiceStarter.java:28)
Caused by: java.net.BindException: Address already in
use at java.net.PlainSocketImpl.socketBind
(Native Method) at
java.net.PlainSocketImpl.bind(Unknown Source) at
java.net.ServerSocket.bind(Unknown Source) at
java.net.ServerSocket.<init>(Unknown Source) at
```

```
java.net.ServerSocket.<init>(Unknown Source) at
sun.rmi.transport.proxy.RMIDirectSocketFactory.
createServerSocket(Unknown Source) at
sun.rmi.transport.proxy.RMIMasterSocketFactory.
createServerSocket(Unknown Source) at
sun.rmi.transport.tcp.TCPEndpoint.newServerSocket
(Unknown Source) ... 9 more
2006-03-10 12:57:17,799 ERROR [main] RMIService(142) -
Failed to start RMI
Connector server. java.io.IOException: Cannot bind to
URL [rmi://abonner-sun:9999/agent]:
javax.naming.NameAlready BoundException: agent
[Root exception is java.rmi.AlreadyBoundException:
agent]
```

*Explanation*
Only one UAF server can run on a computer.

*Action*
Reinstall the RMS into the existing UAF installation.

## ASA Replication Agent Connection Failure

The Adaptive Server® Anywhere (ASA) Replication Agent (`dbltm.exe`) fails when the Replication Monitoring Services (RMS) attempts to establish a connection.

*Explanation*

The RMS uses jConnect™ for JDBC™ and therefore causes `dbltm.exe` to fail when RMS connects to determine its status.

*Action*
Apply the following EBF to `dbltm.exe`: ASA v8.0.3 EBF(5369), ASA v9.0.2 EBF(3272) and ASA v10.0.0.

# Stable Queues

Stable queues temporarily store messages passed by the replication system. Examining the stable queue dump can help determine the cause of a replication system error.

Stable queues are composed of segments stored on disk partitions. Each segment has a megabyte of message space, divided into 64 blocks of 16KB. Segments store messages passed by the system. Each block in a segment corresponds to a physical data transfer between disk and memory. A segment is allocated to only one queue—different queues cannot share segments. Each block within a segment may store one or more messages. A map of the system segments is stored in the `rs_segments` table of the Replication Server System Database (RSSD).

Replication Server deletes messages from queues on a segment basis. Internally, it deletes blocks from the segments. However, the freed space does not appear in the system tables until all of the blocks contained in the 1MB segment are deleted.

These are some of the stable queue commands you can use:

- **sysadmin dump_queue** – dumps the contents of the transaction cache of an inbound or outbound stable queue.
- **sysadmin sqt_dump_queue** – dumps the contents of the transaction cache of an inbound queue.
- **sysadmin dump_file** – specifies an alternate log file to be used by sysadmin dump_queue or sysadmin dump_queue.
- **sysadmin purge_all_open** – purges all open transactions from an inbound queue.
- **sysadmin purge_first_open** – purges only the first open transaction from an inbound queue.

When a stable queue is dumped, a segment may contain deleted blocks along with the undeleted blocks. You can identify the undeleted blocks of a segment by checking the `First Seg.Block` column in the **admin who, sqm** output for the queue.

Sometimes queues are dumped while Replication Server is actively adding and deleting segments from the queues. At such times, the output of the **admin who, sqm** command may not accurately represent the contents of the stable queues. Segments may be deleted before you can enter the **sysadmin dump_queue** command. To prevent Replication Server from changing a stable queue while you examine its contents, you can suspend the database connection or route that uses the stable queue. You can also start the Replication Server in standalone mode to prevent changes to a stable queue. See the *Replication Server Administration Guide Volume 2*.

For information about the stable queue commands, see *Replication Server Reference Manual*.

## Using Traces to Print Commands

Set traces to print the actual commands sent to replicate databases.

The outbound queue for the replicate database contains default commands. If you have created function strings in addition to the defaults, use a special Data Server Interface (DSI) trace flag to print the actual commands that are being sent to replicate databases.

1. Run the debug replication server, **repserver.diag**, to set traces.

   By default, **repserver.diag** is located in the `bin` directory.
2. Forward the output to the standard output or to the error log:

   - Standard output:
     ```
     trace=GEN,TRC_STDERR
     ```
   - Error log:
     ```
     trace=GEN,TRC_ERRLOG
     ```
3. Dump the commands that are sent to the replicate site:
   ```
   trace=DSI,DSI_CMD_DUMP
   ```

   Commands are logged in this form:
   ```
   Command sent to 'RDS.RDB' : command
   ```

## Confirming Suspected Problems

Use **sysadmin dump_queue** to confirm that a suspected problem exists.

1. Log in to the Replication Server.
2. Execute **admin who, sqt**.
3. Use the output from **admin who, sqt** to determine which inbound queue has an open transaction.

   Open transactions appear with the state marked "st:O". The `Info` column of this row holds the queue number and type. The transaction state is followed by the number of commands and then a local queue ID for the **begin transaction** command in the form `segment:block:row`.

   In the following example of **admin who, sqt** output, the transaction has a queue number of 103 and a queue type of 1, is in an open state and consists of three commands. The **begin transaction** command is located in segment 21, block 28, row 0:
   ```
   103:1 st:O,cmds:3,qid:21:28:0
   ```
4. Dump block 28 of segment 21:
   ```
   sysadmin dump_queue, 103, 1, 21, 28, 1, RSSD
   ```

Replication Server writes the text of the commands in block 28 of segment 1 into the `rs_queuemsg` and `rs_queuemsgtxt` system tables or into the Replication Server log.

5. If you used the **RSSD** option, view the command information found in `rs_queuemsg`:

```
select * from rs_queuemsg
  where q_number = 103 and q_type = 1 and
  q_seg = 21 and q_blk=28 and q_row = 0
```

6. Execute a similar query against `rs_queuemsgtxt` to obtain the text of the command.

# Dump Queue Output Interpretation

Ilnterpret the **sysadmin dump_queue** command output on various types of queues.

Stable queues consist of partitions. Partitions are created with the **add partition** command. Whenever a partition is added to the replication server, it is divided into 1 MB segments. Segments are divided into 64 blocks of 16 KB. Messages have fixed-size headers and variable-size messages that are written in blocks.

## Example 1: Outbound Queue after create subscription

Interpret an outboud queue containing a **create subscription** transaction.

The **create subscription** command at the replicate Replication Server starts the materialization process by selecting from the primary database:

```
create subscription emp_queue_sub
for emp_queue
 with replicate at NYDS.nydb
```

*Queue Dump*

Dump the entire queue:

```
sysadmin dump_queue, 103, 0,-1,1,-2
```

Output:

```
I. 2006/07/05 08:20:20. QUEUE DUMP FOR 103:0
I. 2006/07/05 08:20:20. BLOCK BEGIN
q_number=103 q_type=0 blk=0:1 cnt=1
I. 2006/07/05 08:20:20. Begin Transaction
Origin User= Tran Name=
I. 2006/07/05 08:20:20. ENTRY ver=1100 len=188 orig=102
lorig=0 oqid=000000000000016c7000004ff0004000004ff0000
000097f50087a0720000000000000000 lqid=0:1:0 st=21
tr='00000000000016c70000' NYDSnydb comlen=69
activate subscription 0x0100006580000065 0
I. 2006/07/05 08:20:20. END QUEUE DUMP FOR 103:0
```

*Explanation*

- BLOCK BEGIN
  The beginning of a 16K block.
- q_number=103
  The queue ID.
- q_type=0
  The queue type (0 = outbound, 1 = inbound).
- blk=0:1
  Block identifier (segment 0: block 1).
- cnt=1
  The number of entries (commands) in this block.
- Begin Transaction Origin User= Tran Name=
  Beginning of a transaction with originating user and the transaction name. *Tran Name* is empty when the original transaction at the primary database was explicitly started with a **begin transaction** without supplying a transaction name.
- ENTRY
  The actual message.
- len=188 orig=102
  The message length and the originating site ID.
- oqid=00000000000016c7000004ff0004000004ff0000000097f50087a072 0000000000000000
  The origin queue ID: the queue ID assigned to the command string at the place the command originated from. This ID is used for duplicate detection.
- Lqid=0:1:0
  Local queue ID: Commands logged by an SQM are assigned a local queue ID. This queue ID is sent to the next queue as the sender queue ID and is used by the destination to acknowledge the command. *Lqid* consists of segment number, block number and row number. Row numbers start at 0.
- St=21
  Status and command type. The field is a bitmask which has status and command type set. This bit's settings are 0x01 and 0x20, a commit and materialization command.
  0x01 = status of a commit
  0x02 = status of a rollback
  0x04 = status of a begin
  0x08 = status of a orphan statement
  0x10 = status of a materialization statement
  0x20 = status of a materialization statement
  0x40 = status of a tran going to primary
  0x80 = ignore duplicates for DDL commands
  0x0100 = status of a materialization command

0x0200 = dump/load message
0x0400 = routing message
0x0800 = flag for DDL generation
0x1000 = DDL generation at DSI flag
0x2000 = special SQM message
0x4000 = SQT mini-abort
0x8000 = message has been zapped

- tr= '00000000000016c70000' NYDSnydb comlen=69 activate subscription 0x0100006580000065 0
  Transaction ID and command, *comlen* indicates the length of the command.

## Example 2: Inbound Queue after a Series of Commands

Interpret an inbound queue containing different transactions.

These commands were executed at a primary database, `NYDS.nydb`:

```
sp_setreplicate emp_queue, true
insert emp_queue
   values("123456789", "Davis", "Gen",
          "Process Engineer","1/1/95", 1111)
insert emp_queue
   values("987654321", "Irvine", "Ben",
          "Microwave Engineer","3/5/94", 2222)
grant all on emp_queue to public
```

### *Queue Dump*

Dump the inbound queue of the primary Replication Server:

```
sysadmin dump_queue, 102,1,-1,1,-2
```

Output:

```
I. 2006/07/05 08:27:05. BLOCK BEGIN
q_number=102 q_type=1 blk=0:4 cnt=10
I. 2006/07/05 08:27:05. Begin Transaction
Origin User=sa Tran Name=rs_logexec
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=216 orig=102
lorig=0 oqid=00000000000016c7000004ff0005000004ff0005
000097f5008af20b0000000000000000 lqid=0:4:0 st=4
tr= '00000000000016c70005' NYDSnydb comlen=97
begin transaction
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=204 orig=102
lorig=0 oqid=00000000000016c7000004ff0007000004ff0005
000097f5008af20b0000000000000000 lqid=0:4:1 st=1
tr= '00000000000016c70005' NYDSnydb comlen=86
commit transaction
I. 2006/07/05 08:27:05. Begin Transaction
Origin User=sa Tran Name=_ins
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=208 orig=102
lorig=0 oqid=00000000000016c7000004ff0008000004ff0008
000097f5008af20b0000000000000000 lqid=0:4:2 st=4
tr= '00000000000016c70008' NYDSnydb comlen=91
begin transaction
```

```
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=524 orig=102
lorig=0 oqid=00000000000016c7000004ff0009000004ff0008
000097f5008af20b0000000000000000 lqid=0:4:3 st=2097152
tr= '00000000000016c70008' NYDSnydb comlen=406
insert into dbo.emp_queue
 (emp_id, emp_first, emp_last, emp_title, emp_date, id)
 values ('123456789', 'Gen', 'Davis',
          'Process Engineer', '1/1/95', 1111)
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=204 orig=102
lorig=0 oqid=00000000000016c7000004ff000a000004ff0008
000097f5008af20b0000000000000000 lqid=0:4:4 st=1
tr= '00000000000016c70008' NYDSnydb comlen=85
commit transaction
I. 2006/07/05 08:27:05. Begin Transaction
Origin User=sa Tran Name=_ins
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=208 orig=102
lorig=0 oqid=00000000000016c7000004ff000b000004ff000b
000097f5008af20b0000000000000000 lqid=0:4:5 st=4
tr= '00000000000016c7000b' NYDSnydb comlen=91
begin transaction
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=528 orig=102
lorig=0 oqid=00000000000016c7000004ff000c000004ff000b
000097f5008af20b0000000000000000 lqid=0:4:6 st=2097152
tr= '00000000000016c7000b' NYDSnydb comlen=409
insert into dbo.emp_queue
 (emp_id, emp_first, emp_last, emp_title, emp_date, id)
 values ('987654321', 'Ben', 'Irvine',
 'Microwave Engineer', '3/5/94', 2222)
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=204 orig=102
lorig=0 oqid=00000000000016c7000004ff000d000004ff000b
000097f5008af20b0000000000000000 lqid=0:4:7 st=1
tr= '00000000000016c7000b' NYDSnydb comlen=85
commit transaction
I. 2006/07/05 08:27:05. Begin Transaction
Origin User=sa Tran Name=_grrev
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=212 orig=102
lorig=0 oqid=00000000000016c7000004ff000e000004ff000e
000097f5008af20b0000000000000000 lqid=0:4:8 st=4
tr= '00000000000016c7000e' NYDSnydb comlen=93
begin transaction
I. 2006/07/05 08:27:05. ENTRY ver=1100 len=204 orig=102
lorig=0 oqid=00000000000016d2000005010016000004ff000e
000097f5008af20b0000000000000000 lqid=0:4:9 st=1
tr= '00000000000016c7000e' NYDSnydb comlen=85
commit transaction
```

*Explanation*

- Q_type=1
  An inbound queue.
- Tran Name=rs_logexec
  Name of the transaction **sp_setreplicate**.
- Tran Name=_ins

Adaptive Server uses transaction name _ins for implicit insert transactions.

- Tran Name=_grrev

  The **grant** and **revoke** permission commands get a special transaction name from the Adaptive Server.

**Note:** You see all **begin**, **commit**, and **rollback** commands, regardless of whether or not the transactions contain changes that need to be replicated. When the RepAgent reads the log, it does not know whether or not the transaction includes changes to objects that are marked for replication.

## Example 3: Inbound Queue after update

Interpret an inbound queue containing an **update** transaction.

This **update** command was executed at the primary database:

```
update emp_queue
   set emp_first = "General"
   where emp_id = "123456789"
```

### *Queue Dump*

Dump the inbound queue of the primary Replication Server:

```
sysadmin dump_queue, 102,1,-1,1,-2
```

Output:

```
I. 2006/07/05 08:29:52. BLOCK BEGIN
q_number=102  q_type=1 blk=0:5 cnt=3
I. 2006/07/05 08:29:52. Begin Transaction
Origin User=sa Tran Name=_upd
I. 2006/07/05 08:29:52. ENTRY ver=1100 len=208 orig=102
lorig=0 oqid=00000000000016e300000502000020000005020002
000097f5008bedfe0000000000000000 lqid=0:5:0 st=4
tr= '00000000000016e30002' NYDSnydb comlen=91
begin transaction
I. 2006/07/05 08:29:52. ENTRY ver=1100 len=568 orig=102
lorig=0 oqid=00000000000016e300000502000040000005020002
000097f5008bedfe0000000000000000 lqid=0:5:1 st=2097152
tr= '00000000000016e30002' NYDSnydb comlen=451
update dbo.emp_queue
   set emp_id='123456789', emp_first='General',
       emp_last='Davis', emp_title='Process Engineer',
       emp_date='1/1/95', id=1111 where id=1111
I. 2006/07/05 08:29:52. ENTRY ver=1100 len=204 orig=102
lorig=0 oqid=00000000000016e300000502000050000005020002
000097f5008bedfe0000000000000000 lqid=0:5:2 st=1
tr= '00000000000016e30002' NYDSnydb comlen=85
commit transaction
```

### *Explanation*

- Tran Name=_upd

Adaptive Server uses transaction name “upd” for implicit update transactions. RepAgent reads this name from the log.

## Example 4: Outbound Queue after update

Interpret an outboud queue containing an **update** transaction.

An update has been performed at the primary database. The minimal column feature has not been set for the replication definition:

```
update emp_queue
   set emp_first = "General"
   where emp_id = "123456789"
```

*Queue Dump*

Dump the outbound queue or the replicate:

```
sysadmin dump_queue, 103, 0, -1,1,-2
```

Output:

```
I. 2006/07/05 08:31:32. BLOCK BEGIN
q_number=103 q_type=0 blk=0:3 cnt=3
I. 2006/07/05 08:31:32. Begin Transaction
Origin User=sa Tran Name=_upd
I. 2006/07/05 08:31:32. ENTRY ver=1100 len=204 orig=102
lorig=0 oqid=00000000000016e300000502000500000 5020002
000097f5008bedfe0000000000000001 lqid=0:3:0 st=4
tr='00000000000016e30002' NYDSnydb comlen=88
begin transaction
I. 2006/07/05 08:31:32. ENTRY ver=1100 len=268 orig=102
lorig=0 oqid=00000000000016e300000502000500000 5020002
000097f5008bedfe0000000000000002 lqid=0:3:1 st=0
tr= '00000000000016e30002' NYDSnydb comlen=149
update emp_queue
   set emp_id='123456789', emp_first='General',
       emp_last='Davis', emp_title='Process Engineer',
       emp_date='1/1/95', id=1111
   where id=1111
I. 2006/07/05 08:31:32. ENTRY ver=1100 len=180 orig=102
lorig=0 oqid=00000000000016e300000502000500000 5020002
000097f5008bedfe0000000000000003 lqid=0:3:2 st=1
tr= '00000000000016e30002' NYDSnydb comlen=63
commit transaction
```

*Explanation*

- cnt=3

  There are 3 rows in this block.
- Tran Name=_upd

  If you perform an **update** and did not explicitly start a transaction, Adaptive Server uses _upd as the transaction name. Likewise, Adaptive Server uses _del for a **delete** command and _ins for an **insert** command.

- st=4, st=0, st=1
  Status for **begin**, **other**, and **commit**.

# Language, Sort Order, and Character Set Issues

Errors may occur when you use localization features.

Replication Server provides support for international environments with:

- Localization of messages into English, French, German, Spanish, Chinese, Korean, and Japanese languages
- Support for all Sybase-supported character sets, with character set conversion between Replication Server sites
- Support for nonbinary sort orders

Because Replication Server works in a distributed environment with many independent processes, you must set up its languages, sort orders, and character sets carefully to avoid problems. Follow the recommendations in the *Replication Server Design Guide*.

**Note:** RepAgent uses the same language, sort order, and character set settings as the Adaptive Server.

## Message Language Problems

You can configure Replication Servers to print messages to the error log and to clients in English, and other foreign languages. This allows server programs in your replication system, including Replication Servers, and Adaptive Servers, to write messages to their error logs in the configured language. Depending on the server, they may or may not send messages to a client in the client's language.

For example, Adaptive Server checks for the language setting of its client (Replication Server) and returns messages in that language. Replication Server, however, does not check for a client's language; instead, it returns messages to a client in its own language. This can result in error logs with messages in different languages if the servers are configured in different languages.

Stored procedures use the language of their Adaptive Server connection. Therefore, if you have a mixed-language environment, Adaptive Server stored procedure messages may be in a different language from the Replication Server.

For example, if you log in to Adaptive Server using French and execute a stored procedure, the stored procedure generates messages in French, even if the language setting of Replication Server is English or Japanese. If you log in to Adaptive Server with a language that is not installed at the Adaptive Server, Adaptive Server returns stored procedure messages in English.

Sybase may not always translate new or changed messages into French, German, Spanish, Chinese, Korean, or Japanese. In this case, the Replication Server, or **rs_subcmp** messages use the English version of the message.

Misplaced message localization files can also cause message language problems. Verify that the files are in the correct directories.

To avoid the confusion that can result from a mixed-language error log, Sybase recommends that you configure the Replication Server so that all languages are the same.

## Sort Order Problems

Most sort order problems in Replication Server are a result of different sort orders being used in different servers throughout your system.

- You receive a Data Server Interface (DSI) error when trying to insert or update a record into a table with a unique index on a character column. Under the primary data server's sort order, the value is unique; but under the replicate data server's sort order, it is not unique. For example, "lvis" and "LVIS" are considered distinct under a binary sort order but are considered equal under the `nocase` sort order.
- A **drop subscription with purge** command does not completely remove the rows at the replicate database. A query to select records to purge misses some records under the replicate data server's sort order.
- Replicated data is not what the replicate site expects.

  When this occurs, a subscription probably has one or more character clauses. Updates during the lifetime of a subscription are processed with the primary Replication Server's sort order, leading to unexpected semantics at the replicate data server if the sort orders differ.

Correct these problems by editing the Replication Server configuration file to use the same sort order and, if necessary, by running **sybinit** to make the primary data server and replicate data server use this sort order as well.

### Sort Order

The sort order, or collation sequence, used by a server determines how character data and identifiers are compared and ordered. A sort order is determined by the character set.

Replication Server supports all Sybase-supported sort orders, including nonbinary sort orders. Nonbinary sort orders are necessary for the correct ordering of character data and identifiers in European languages.

Use a nonbinary sort order if you have both:

- Differing character sets between the primary and replicate data servers, and
- Data with 8-bit characters,

and you intend to do either of the following:

- Include columns with this data in the **where** clause of a subscription, or
- Query the database with an **order by** clause involving the columns with this data.

## Character Set Problems

Character set problems can arise when servers have different or incompatible character sets.

These problems can be corrected by configuring all servers in your replication system to use compatible character sets.

- 8-bit or multibyte data is corrupted when replication is taking place between a multibyte and a single-byte data server.

  Sybase does not support any character set conversion between single and multibyte systems; it simply passes the data and object names as is. Since 8-bit and multibyte characters are character-set-specific, their semantics change after replication.
- Character data in the replicate data server has some question marks (?) substituted for non-ASCII characters, even though the primary and replicate data servers use compatible character sets. Also, object names originating at a remote site have some questions mark (?) characters.

  This happens because different character sets, even compatible ones, have some unique characters. When trying to convert a character that does not exist in the replicate data server's character set, a question mark (?) is substituted for the unrecognized character.

  If you are replicating between compatible character sets (for example, iso-1 and cp850), make sure that your object names and character data do not include any of the characters that are not common to both character sets.

## Language and Globalization

There is a limitation when using Japanese character sets in Replication Server. Neither the eucjis nor the sjis character set can be converted; this issue affects both Adaptive Server and Open Client™ and Open Server libraries.

### *Hankaku Katakana Conversion*

In general, Japanese character sets are compatible. However, Hankaku Katakana characters, although they exist in both the eucjis and sjis character sets, cannot be converted. Converting data that contains Hankaku Katakana characters between eucjis and sjis does not work. This conversion problem occurs with `character` datatypes and the `text` datatype and is documented in the *Adaptive Server Enterprise System Administration Guide Volume 1 > Configuring Client/Server Character Set Conversions*.

This conversion problem affects both Adaptive Server and the Sybase Open Client and Open Server libraries. Because Replication Server uses these libraries for all conversions, this problem also affects Replication Server.

In Replication Server, this type of failure is treated in the same way as is the case of a single character missing from the target character set. The remainder of the conversion succeeds and replication proceeds, and problem characters are replaced by question marks in the target data area. There is currently no way to escape this restriction with the Sybase connectivity libraries. However, in Adaptive Server, if you turn on trace flag number 2402, you can remove this restriction.

### *Using Trace Flag 2402*

Generally, Sybase recommends that you set up your replication system so that Replication Server handles all character set conversions at the replicate Replication Server and prevents the replicate data server from performing any conversions. In this case, you can work around the Hankaku Katakana restriction if you set up your system so that the replicate data server performs the conversion.

This table shows how this might look if the primary data server used the sjis character set and the replicate data server used eucjis. Communication in this system is between each data server and its Replication Server and between the two Replication Servers.

| | |
|---|---|
| Primary Replication Server | sjis |
| Replicate Replication Server | sjis |
| Primary data server | sjis |
| Replicate data server | eucjis |

The primary and replicate Replication Servers are configured to use the same character set as the primary data server. (If only one Replication Server manages the primary and replicate data servers, configure it with the character set of the primary data server.)

In this configuration, when the replicate Replication Server connects to the replicate data server with character set sjis, the replicate data server detects this condition and converts data into its own character set, eucjis. If trace flag 2402 is activated in the replicate data server, then the conversion includes the Hankaku Katakana characters.

### *Setting Up Workaround*

1. Configure your system as suggested.
2. Turn on trace flag 2402 in the replicate data server (Adaptive Server) by including **-T2402** on the command line when you start Adaptive Server.

### *Changing Default Date Format for a Language*

If you modify the `common.loc` file to change the default date format for a given language, make the corresponding change to the `syslanguages` table on all affected Adaptive Servers.

# Index

## E



## F

## G



## H



## I



## K



## L



## M



## N



## O

## P



## Q



## R

## S

## T



## U



## V



## W